doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

# Modelling and simulating organizational ransomware recovery: structure, methodology, and decisions

Marius-Constantin Ilau,[1] Adrian Baldwin,[2] Tristan Caulfield,[1] * and David Pym[1,3,4]

[1]Department of Computer Science, UCL, Gower St., London, WC1E 6BT, England, U.K., [2]HP Security Lab, HP Inc., 1 Redcliffe St., Bristol, BS1 6NP, England, U.K., [3]Institute of Philosophy, University of London, Malet St., WC1E 6HU, England, U.K. and [4]Department of Philosophy, UCL, Gower St., London, WC1E 6BT, England, U.K.
*Corresponding author: t.caulfield@ucl.ac.uk
FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

## Abstract

The problem of maintaining organizational resilience in the face of ransomware attacks has become an important issue for organizations. Organizational networks and IT infrastructure have become increasingly complex, and it is often unclear how decisions about technology, policy and recovery strategy will impact resilience. In this context, the paper focuses on two primary objectives. First, to offer security decision-makers a way of better understanding the impact of deploying different recovery solutions at organizational level by means of simulation modelling and comparative analysis of solutions. Second, to illustrate the suitability and benefits of using semantically justified, compositional system models together with a rigorously defined co-design model-construction methodology, in a complex scenario. Our choice of organizational recovery as modelling target is motivated through both form and complexity, allowing for illustrating the model conceptualization and construction methodology in a sufficiently rich context. We conceptualize the ransomware behaviour, organizational structure, IT infrastructure and recovery choices and behaviour based on literature surveys and expert knowledge. Then, construct a modular, simulation model representing a generic target organization using our co-design approach. We execute the model over 9000 different parameter configurations, totalling an amount of 450000 iterations. We analyse the results, both in three specific scenarios deemed organizationally relevant and at the general level — through sensitivity analysis — and, exemplify possible ways in which the model can help inform decision-makers about their possible recovery choices.

**Key words:** distributed systems, ransomware, recovery, organizations, modelling methodology, simulation, compositionality, interfaces

## Introduction

System modelling can allow decision-makers to explore choices when deploying or adapting different solutions. In this paper, we illustrate our mathematically rigorous, semantically justified, compositional system modelling approach, together with an associated co-design-based model-construction methodology, and show that it is sufficiently rich to aid decision-makers with practical security problems. Our system models are grounded in an abstraction of distributed systems and are handled mathematically using basic ideas from abstract algebra and process algebra. Our co-design methodology builds on the classical mathematical modelling cycle in order to incorporate explicit interaction, in a 'translation zone', between modellers and other stakeholders. Throughout the paper, we use an example of organizational recovery to demonstrate the modelling approach and methodology and show the insights it brings to aid decision-makers. We choose the example of organizational recovery as it is a timely and significant challenge with a form, complexity, and scale that conveniently lends itself to illustrate both the conceptual components of our modelling approach and the methodology of constructing models.

This process of organizational recovery can be defined as the totality of actions an organization can undertake to restore its functions after a traumatic event. From an information security perspective, this can be specified as the ability to restore the integrity and availability of data and services. Organizational recovery represents the main facilitator of organizational resilience. As described by Gibson & Tarrant in (59) and formalized by Ioannidis et al. in (74), organizational resilience should be conceptualized as an ability to adapt to changing organizational circumstances. From a security perspective these changes arise from differing IT usage patterns, the changing landscape of malware attacks as well as failures in IT processes. A wider view would include other phenomena like natural disasters, poor financial and human resource management or the inability to comply with new legislative initiatives.

In this paper, we focus on studying endpoint device recovery mechanisms that enable recovery after failures and attacks. As might be expected, multiple approaches to recovery exist and we give an overview of some of these in Section 2.2. The decision problem here is what mix of mechanisms and supporting IT processes needs to be put in place to have robust and timely recovery strategy. The effectiveness of these approaches depends on the nature of the attack, spread, size, required recovery time, cost of deployment, existing policies, organizational structure, and employee knowledge and behaviour. Our modelling approach leads to the construction of simulation models allowing different choices and outcomes, such as time to recover, to be explored.

We motivate the organizational recovery problem by exploring ransomware attacks. As shown in Richardson & North (123), O'Kane et al. (111) and Oz et al. (112), ransomware on its own represents a constantly evolving threat with serious implications for organizations today. Furthermore, threat assessment reports from government and law enforcement agencies such as (51), (53) or (62) conclude that ransomware is one of the most relevant threats for organizations; Tweneboah et al. (130) gives a more nuanced position on the variations across different industries. In their empirical ransomware research study, Connolly et al. (141) show that out of 55 different ransomware attacks on organizations of different sizes and industries, 21% have managed to fully disrupt business continuity for two weeks or more and in 19% of the cases, organizational recovery took several months if at all. Ransomware provides a rich attack space as it is distributed through a wide range of mechanisms and from a modelling methodology perspective, we show how a varied ransomware attack space can be abstracted and captured in different attack models that can be composed into our overall organization.

We employ a simulation modelling approach (29) underlined by a distributed systems metaphor and co-design construction methodology (30; 25; 10) in an attempt to better understand the impact of different combinations of recovery mechanisms on different types of organizations. Furthermore, we show how our methodology can be used to understand the implications of organizational recovery choices: we illustrate how a model can be constructed, fitted to an organizational structure and produce useful insights. We, therefore, demonstrate the modelling approach using the recovery problem as a large-scale and critical problem area.

In addition to different preferences, different organizations will have different requirements, structure, and architectures. The number of employees, their travel patterns, the size and number of offices, the devices used, the value of the information on different devices, and the network structure will all vary between organizations. We use a compositional modelling approach to provide the flexibility required to create models that can be adapted to capture all these differences between organizations. We discuss how the model and modelling approach allows us to adapt the parameters of the model to fit different these different organizational characteristics. We then show how decision makers can try different recovery and IT process resourcing choices through the model to explore how the choices effect key operational criteria such as speed to recover. In doing so we demonstrate the practicality and the usefulness of the approach for strategic decision making.

In Section 2, we describe the nature of modern ransomware starting from four different behavioral traits manifested in the wild and construct a conceptual setting for the organizational environment and types of recovery techniques. Section 3 is concerned with the underlying elements of the modelling approach, namely the distributed systems metaphor, including theoretical considerations, their interpretation in Julia and a pragmatic discussion on the stochastic nature of the process. In Section 4 we focus on the actual methodology used for the construction and interpretation of the model content and architecture. In Section 5, we present the recovery model and then Section 6 focuses on the construction of scenarios that illustrate possible real-life situations, the parameter selection for such scenarios and addressing validation concerns. Particularly, subsection 6.2 is concerned with the overall model results, interpretation and sensitivity analysis. Subsection 6.6 illustrates possible uses of the model as a tool for supporting security decision-making.

It is our belief that such an approach, based on a rigorous mathematical foundation, can be used to help organizations understand the problem and the consequences of different recovery choices, and help them make better decisions about this challenging problem.

## Background

### Ransomware

Ransomware, as the name suggests, represents a subdivision of malware that is primarily being used to obtain benefits from a target by limiting the target's access and control over information and/or essential operational infrastructure. Such benefits can vary, from traditional cyber-crime motivators like economic revenue (often in the form of cryptocurrencies given the complexity involved in tracing and identifying beneficiaries of such transactions (86)) to more obscure ones, such as gaining competitive advantage economically, politically or militarily by crippling the operational capabilities of the target.

Since ransomware observed in the wild is constantly evolving, we attempt to identify a subset of characteristics that can enable thinking and reasoning about ransomware at a more general level. This represents an initial stage in the conceptualization of the phenomenon and environments under study, as explained in (28), and will further influence both representation and parameter selection procedures at the level of the simulation model presented in Section 4.

#### *Nature of the Problem*

As previously stated, ransomware can be viewed as the instantiation of a coercive action with visible effects at the level of information systems. As described by Schelling (124) and Pape (113), a coercive action compels a party to act involuntarily through means of either threats or force. Most ransomware adheres to this definition because once a target system is infected, either its operability is drastically reduced via actions such as overlaying various windows over visual interfaces, disabling I/O devices or simply interrupting the operating system booting process by displaying a notification, or, the information on the system is directly encrypted to a certain extent. Although targeting different assets, these two different types of ransomware which are sometimes called **locker** and **crypto-locker** essentially fall into the category of coercive actions, because their disseminators (83)

(21) expect to obtain benefits from their victims by the use of force. Information located on systems affected by these types of ransomware can usually be restored by either removing the processes causing the reduced operability (by using an anti-malware tool or reinstalling the operating system ), using a decryption key, exploiting certain implementation vulnerabilities in the ransomware code or simply paying the ransom while accepting the risk of back-doors. However, in the case of crypto-lockers, recovering the information can be non-trivial, particularly in the case of ransomware based on hybrid encryption with large keys.

Ransomware can employ different approaches to maximize its effectiveness. For example, the Reveton ransomware interrupts the boot process and displays warnings that appear to be from government agencies, to try and increase the likelihood of ransom payment (96). As shown in (85) and (102), even though Reveton was not based on hybrid encryption, its financial impact was as high as $50,000 a day or $400,000 per month during its peak period around 2012-2014. Reveton used several different distribution methods: exploit kits such as BlackHole (129) or Cool (70) that used the CVE-2012-1723 Java vulnerability, but distribution through phishing or application downloads on mobile devices (52) was also possible. Furthermore, Reveton was occasionally deployed alongside other malware, such as the Citadel (96) or Zeus (136) trojans that harvested credentials, monitored web traffic, altered HTML code displayed in browser and introduced the infected device into botnets.

Based on strategies to increase the coercive force of the attack, a third category of ransomware can be specified: **leakware**. Generally, malware in this category does not affect the operability of its target systems. The primary goal of such attacks is to obtain sensitive information — common targets include intellectual propriety, third-party information or information that might be deemed as embarrassing — and then threaten the victim with publication in case the attacker's demands are not met. In more abstract terms, this strain of ransomware affects the degree of control that a user has over information, shifting the primary focus of the coercive action from usability or ability to use systems to restraining the possible spread of sensitive data. However, compared to lockers, ransomware in this category are harder to detect because they do not perform operations that reduce the operational capacity of the target. Although they might present automatic spreading capabilities, the actual data exfiltration operation might be performed manually, similarly to the Grozio Chirurgija (65) cosmetic surgery data breach.

Nevertheless, pure leakware attacks did not reach the popularity of lockers, mostly because the extortion tactic they employ can also be used in composition with the forceful coercion of lockers for a greater impact on the victims. This type of double extortion approach can be seen in newer ransomware strains such as Maze(79), Conti(131) or DarkSide(107) which have been the cause of a series of targeted attacks in the near past: the Maze infection of Allied Universal(3) in 2019, the Conti attacks on JVCKenwood (2) 2019 and Ireland's Health Service (105) (71) in 2021 or the Darkside attack on the US Colonial Pipeline (5) (68) in 2021, with the interesting aspect that in the Colonial Pipeline incident, the Darkside group attempted a triple extortion tactic by threatening with additional denial-of-service attacks in case the ransom was not paid.

Last but not least, a fourth category of ransomware has become more and more prevalent in recent years: **destructive ransomware**. When compared to the other three categories, this type of malware no longer focuses on obtaining benefits from victims via coercive actions. Destructive ransomware directly inflicts irreparable damage to an information system by deleting, overwriting or encrypting both user and system files and memory regions. Therefore, instead of coercing victims, disseminators of this malware type usually conceptualize the loss sustained by the victim as an actual gain and are driven by political motivations, so a direct way of recovering the information is usually not integrated in the malware design process. The Shamoon (45) (23) infection of Saudi Arabia and Qatar's national oil companies in 2012 can be seen as a relevant example of this type of attack.

To accentuate the political motivations, similar types of malware have been attributed to the current Russian invasion of Ukraine (95) (36), but only a single variant — Hermetic Wiper (63) — was distributed using worm-like spread capabilities.

Furthermore, we describe two additional mechanisms for disseminating ransomware: via human operation and ransomware-as-a-service. As illustrated in Microsoft's latest security best practices report (43) from June 2022, **human-operated ransomware** represents 'an active attack by cybercriminals that infiltrate an organization's on-premises or cloud IT infrastructure, elevate their privileges, and deploy ransomware to critical data.' and directly focuses organizations rather than singular devices. These types of attacks behave as shown in the Mitre ATT&CK (7) matrix model. For an example of threat conceptualisation using this approach, see Xiong et. al. (138) Nonetheless, the 2021 attack on the information technology infrastructure company Kaseya (33; 54) by the REvil group can be considered an example of human-operated ransomware: the malicious actor managed to leverage a vulnerability in the proprietary remote monitoring and management software for the VSA Cloud and SaaS servers — which shows reconnaissance has been performed — and disseminated a ransomware payload to both Kaseya and a subset of its clients. The impact of the incident varies across sources, with REvil claiming to have encrypted more than one million systems (33) and Kaseya declaring between 800 and 1500 businesses as being affected (120).

However, the latest years have not only brought specialization efforts in terms of more targeted and sophisticated strains of ransomware, but also an increase in the accessibility of deployment for non-technical users. As described in (49) **ransomware-as-a-service** represents a specialization of the software-as-a-service model: skilled malware writers produce high-quality samples which are then employed by less skilled attackers either via a one-time payment or subscription method to be deployed against certain targets, and everything is done via easy to used web interfaces which sometimes even have user reviews, scoring systems and catchy marketing phrases. For extended reviews, see Keijzer (78), Alwashali et al. (4) or Meland et al. (89) for en economic perspective. To grasp the current degree of evolution, Karapapas et al. (76) even describe a proof-of-concept ransomware-as-a-service model based on IPFS file system and Ethereum blockchain. Nevertheless, from a behavioural perspective, the actual ransomware strains used via ransomware-as-a-service do not differ from other strains. In the future, we expect this model of operation to continue developing and provide users with even more dangerous types that might employ direct handler operation or zero-day exploits.

Although ransomware in the above categories is being used to achieve different objectives, the technological means of achievement are similar and have undergone formalization and generalization attempts over time. For example, Young & Yung describe ransomware in their seminal 1996 article as a 'cryptovirology attack' (140) in which cryptographic approaches are used offensively to 'mount extortion based attacks that cause loss of access to information, loss of confidentiality, and information leakage, tasks

which cryptography typically prevents' (140). Their attack follows a hybrid encryption scheme and resembles the formalization of a digital envelope.

Modelling ransomware does not require understanding only the inner workings of the ransomware code. For example, the identification of a victim could be an automated process of scanning IP ranges, or it might involve a carefully planned reconnaissance procedure. Once identified, the victim could be infected through a series of means: phishing, spear-phishing, physical social engineering to provide access or introduce malicious hardware components into the system, file downloads, Trojans and others. Furthermore, after the initial infection, additional worm-like network spreading (73) might occur via misconfigured network shares — with or without the need for zero day exploits such as EternalBlue (18) — automated phishing based on gathered information from the already infected devices or direct credential harvesting and dropping if escalation of privileges is successful (32) (31). For a complete list of ransomware strains that contributed to the above behavioral classification, see Appendix A, Table 1. Given the evolution of ransomware and the development of modern strains such as Maze or Conti which combine characteristics from multiple categories to increase the coercion on users, a similar compositional capability at the level of modelled malware behavior is required and will be presented in Section 4.

## Recovery

As described at the start of Section 1, the goal of organizational recovery is to bring an organization into a business-as-usual state after a traumatic event. In the context of this paper, this traumatic event is represented by a ransomware outbreak, although the model we present could be used with other parameter configurations to simulate other types of events, such as a flooding denial-of-service attack, for example.

For a better understanding of the possible recovery mechanism choices, we firstly draw attention to a few important aspects about the nature of the threat. Modern ransomware strains are harder to fully erase at an organizational level because of a series of spreading and persistence mechanisms: as seen in Section 2.1, ransomware such as WannaCry, NotPetya or BadRabbit extend the class of crypto-lockers with powerful worm-like spreading mechanisms constructed on top of zero day exploits, greatly increasing their dissemination speed. However, this evolution cannot be considered as a statistical indicator of a higher severity attack on its own yet: as shown in Connolly et al. (141), the hypothessis that 'the crypto-ransomware propagation class influences the impact severity of a ransomware attack' is rejected in their study with the comments that a combination of factors such as the nature of business, availability of resources to recover data or pay the ransom, the type of systems affected and level of preparedness should be further analysed. An insight we consider relevant in their study is that the overall severity of 'generation two crypto-ransomware', which maps onto our crypto-locker category without worm like spreading achieved a score of 0.32, whereas the 'generation three crypto-ransomware' which manifests worm-like spreading only achieved 0.23. We argue that this should not be interpreted as spreading factor not being relevant to the severity of the infection, but rather that the security posture and type of organization play a more important role than purely the technical advancements of the ransomware. Furthermore, one of the most severe attack they identified was a worm-based crypto-locker that targeted a large public organization — GovSecA as named in the study — and managed to encrypt close to 100 servers. At the time of their study, the organization still had not fully completed their recovery processes, almost 8 months later. The example, does not give details about why this recovery period was so significant, but we allow ourselves to speculate that reinfection played a big part. For more information about reinfection, including a stochastic epidemics model that simulates parts of the behavior of Viking.gt malware on Norwegian Bank, see Hole (69).

Nevertheless, it is not only the spreading speed that has evolved over time, but also the persistence mechanisms employed — with techniques such as modifying registry keys, altering run once keys, the bootexecute key, boot helper objects, keys used by the WinLogon process, startup keys, launch additional services to facilitate reinfection in the case of recovery, maintain a command and control structure or avoid detection methods during data exfiltration or altering the DLL search order mechanism itself as being only a handful of approaches. For additional information and platform specific approaches, see (61; 134; 20). In addition to these purely persistence oriented mechanisms, modern ransomware has also exhibited a series of destructive elements such as overwriting the MBR (125) in the case of NotPetya and Shamoon, which make backups and re-imaging of devices mandatory to ensure business as usual can be re-established. The details of the process and how organizations often end up paying ransoms rather than rebuilding their systems are described in (133).

However, it is not just ransomware that spreads over networks that necessitates recovery at scale. Malware families, such as Emotet (35), also have mechanisms to spread rapidly through email. Spreading patterns have evolved to include WiFi (14), and can make it hard to clean corporate systems without taking everything offline. Some malware can be cleaned by Anti-Virus systems, but it can be hard to guarantee and trust that systems are clean; hence, easing the re-imaging process can become an essential part of a company's response to malware and attacks. For example, the SANS incident-responder's handbook recommends re-imaging of systems' hard drives to ensure malware is eradicated (84), with recent surveys showing incident response processes often leading to re-imaging (22).

Companies are becoming aware that they must start planning for both large-scale and smaller-scale outages in order to get their and systems staff back up and functioning as soon as possible (50). There are, of course, various products and approaches to backup, re-imaging and restoration. However, there are a lack of tools to help IT decision-makers decide on the most appropriate strategy and assess whether they have the necessary tools and infrastructure in place. This is the problem that we look at within this paper: we demonstrate how modelling and simulation can be used to aid the decision-makers in the choices they make. A good example of an executive level document which details the Microsoft strategic and operational approach against ransomware can be found in (43) and focuses at least on secure backup, privileged access plan, data protection plan and security posture and governance, all provided with clear accountability paths and KPIs.

There are several approaches that organizations might employ in order to maintain the operation and re-imaging of client systems. Underlying these approaches there are three basic choices:

*Full System Backups*

Some companies will have backup systems that keep a full system backup of each client. Restoration will then happen by reinstalling this full backup. The backup vendor would support this restoration process with a typical reinstall process involving the download of a Windows PE agent along with the full system backup, placing this onto a bootable USB stick, and then, through the BIOS menus, booting into this cut-down version of Windows, which will reinstall from the full backup — for an extensive survey of classical recovery methods, see Chervenak et al. (34). For an updated version, including information regarding cooperative approaches, see Killijian et al. (80). Taking full backups is becoming less common, particularly in the hardware sense, as it means keeping copies of many standard system files and there are advantages to re-imaging to a clean up-to-date OS image.

*Re-imaging to a Corporate Image*

A more common scenario is for companies to have a standard corporate image along with a data backup strategy. For example, a company will often create a Windows image containing corporate management tools — such as a management agent for a system such as Microsoft's Endpoint Configuration Manager — and its security software, both AV and EDR systems, such that when a client image installation happens the system is secure and manageable (122). Microsoft provide a management deployment toolkit (93) that describes and supports this overall deployment process. The management tools will then typically help install other applications as required. Such images will be updated regularly — quarterly or half yearly, say — to include the latest version of Windows, patches, and software.

Data backup may be through backup software to an enterprise server or the cloud, although companies are increasingly using synchronized cloud-based storage such as OneDrive, where data is stored on the cloud with local copies cached on the endpoint.

From a recovery perspective, as a user decides they need to re-image a system they get hold of a bootable image on a USB stick (or occasionally a DVD) and boot into this to re-image the system. In an office environment, where there is IT support, the IT engineers will maintain a set of current OS images on bootable media. In smaller offices, or where there are home workers, the OS image can be downloaded and there will be instructions for the user to create the bootable media and reinstall. Such instructions can be complex for a typical user and require access to a USB stick that can be wiped and reformatted. If a user is at home they would need a functioning computer to use to download the image.

IT support labs will also often have a PXE boot set-up to make re-imaging easier (92). They have an image hosted on a local server and use PXE boot to point the system to that image to install. This can ease the problem of setting up larger numbers of client systems, although it requires staff and infrastructure.

*Modern Management*

There is an increasing move towards the use of modern management systems (Uniform Endpoint Management), such as Microsoft's InTune System (91). This approach allows the use of a standard Windows image, such as that initially placed on the computer, rather than a specially maintained corporate image. During the install process, the management infrastructure will push critical security patches, AV signatures, Windows domain policies (Group policy objects), and necessary software. This produces a similar effect to having a corporate image, but removes the need to maintain custom images.

Typically, after a new image has been installed, an out-of-the-box experience (OOBE) process runs, the user will be lead through configuration screens, and will login using their corporate email. The login directs the system to a cloud-based management server, so that the enterprise configurations can be found and installed, and the computer added to the enterprise domain. This process can be simplified further using Microsoft's AutoPilot (94), where a computer is preregistered as belonging to a company, and user interactions and configurations can be simplified and reduced.

The re-imaging process still requires that the user can get hold of a clean Windows install image. However, the company does not need to maintain and host its own Windows image. Instead, a user can download the latest OS copy from either the PC manufacturer or from Microsoft. Companies using these mechanism will typically use cloud-synced storage, discussed above, to provide data resilience and, as the system is re-imaged and added to the corporate domain, data will gradually be synced back to the client.

*Re-imaging Mechanisms*

Re-imaging will typically involve booting from an ISO image and installing this onto a drive, or via a reduced version of Windows, such as WinRE or WinPE, which can install Windows from WIM files. Windows itself also includes a number of repair processes (90) — for example, allowing rollbacks to previous snapshots using Shadow Volume Copy. However, malware such as ransomware often disables volume shadow copy and delete snapshots, so making recovery and the retrieval of older files hard. Incident responders often recommend a clean install to ensure malware is eradicated.

Re-imaging processes require a boot into a system rather than the normal OS. The boot process is controlled by the BIOS, which will have a defined boot order and set of devices that can be used to boot the system. Many systems will boot from an attached USB device or PXE boot before the main disk, making re-imaging easy — but with no controls. Early in the boot cycle, users can get into the BIOS menu and boot to an alternative device. Some enterprises will lock down the BIOS with passwords and ensure the system boots only from the internal disk and, in this case, re-imaging will require an IT support engineer who knows the BIOS password.

HP has built a bare metal recovery system (HP 'Sure Recover' (72)) into the BIOS in order to simplify the re-imaging process. The Endpoint Security Controller (EpSC) holds a configuration containing the location of image-servers, which may be either HP's servers for standard Windows images or other servers specified by the enterprise. The configuration also contains public keys of the authority allowed to sign the Windows image to be installed and, in this way, an enterprise can guarantee the image being installed has integrity and has been approved. Recovery can be triggered by the user at boot time through the BIOS recovery

option or it can trigger automatically when the system fails to boot — such as with NotPetya. When triggered, the BIOS gets this configuration information and uses it to download a recovery agent, which then downloads the full OS image and re-images the system. Both recovery agents and the full image are signed and the signature is validated as part of the recovery process ensuring authenticity of the recovered image. The process simplifies recovery for the user as they no longer need to be able to find where to obtain the OS image and do not need an available USB stick. From the enterprise perspective, it allows the enterprise to lock the BIOS without support engineers doing rebuilds, as well as guaranteeing that the image installed is correct.

An additional option is available, HP 'Sure Recover' Embedded, which adds additional storage onto the endpoint device that is used to keep a local up to date copy of the recovery image. It reduces network download times and means recovery can happen when no network is available, or when networking is limited or metered.

*Enterprise Recovery Choices*

The descriptions above show that there is a wide range of choices available to the enterprise as it looks to implement an image management and recovery strategy; for example:

- Maintain a corporate image or use a standard image. There is a choice as to how often the image is updated. After recovery, patches will need installing and updating images more often will reduce the need and time taken for post re-image patching. However, this option requires additional resources to manage the actual image, in the sense that its actual content might require frequent updates. When a company maintains its own OS images, they must maintain servers to support the download of images. Download speeds may depend on the location of these servers, or how they are distributed over the world, the location of users, and the network bandwidth available in an office or to a home or travelling user. The volume of traffic to these servers will depend on the recovery scenario — in terms of the numbers of users likely to be recovering at a given time and on the state of the underlying infrastructure.

- How much IT support is needed and how much in each office? Many companies are looking to reduce IT costs, and this often creates pressure to centralize help-desks and remove support from offices. However, a lack of local support and locally kept OS images can delay recovery times for multiple reasons: some users might require physical support in starting the recovery procedure and some might simply needed to be guided remotely, but a bottleneck in the actual support stream can lead to a reduced stream of recovery even if the infrastructure is extremely capable. At the same time, the enterprise will need to plan for remote workers either working from home or as they travel; such support needs have increased dramatically with Covid 19, for example, since the actual work location of employees is less geographically bound to an office.

- Control over the re-imaging process can bring various choices with which the enterprise may wish to lock down its client platforms, but this adds a considerable burden in recovery.

- There are many different data backup strategies, from the use of cloud synced drives through to full system backups. Each will have an impact on the ease of recovery and potential user data loss.

- The different re-imaging techniques, such as using a USB stick or PXE boot, in comparison to having recovery mechanisms such as HP 'Sure Recover' built into the system — whether with an image stored locally or downloaded.

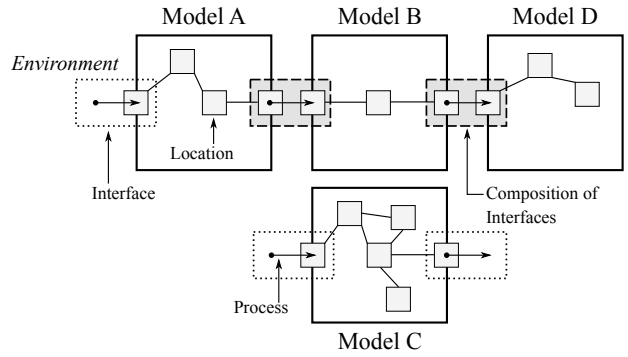



**Fig. 1.** Interfaces, Composition, and Substitution

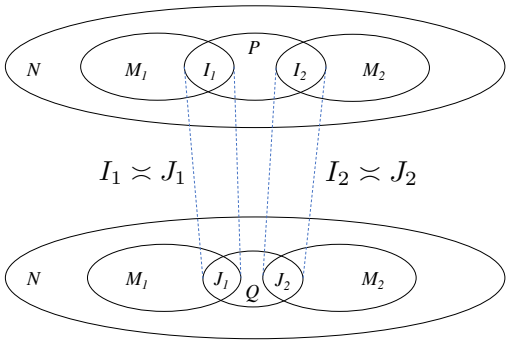



**Fig. 2.** Interfaces and Substitution

## Background on Modelling Methodology

This section describes the modelling methodology we have developed and employ in this work. We first describe how concepts from distributed systems are used as the fundamental components of models. We then give an overview of how these concepts can be represented mathematically, giving a rigorous foundation. Finally, we use a brief example to demonstrate how models can be written and executed using a library we have developed for the Julia programming language.

The growth of interconnected networked systems led to the development of the concept and theory of distributed systems in computer science. This paradigm views systems as collections of components, in different locations, that work together to perform some task and communicate by sending information or messages over network connections.

This view is obviously very specific to its focus on computer systems, but its concepts can be taken more generally to provide a useful metaphor for understanding all types of systems — and ecosystems. There are three key components upon which we draw.

- **Location** — Distributed systems naturally have a concept of different locations, which are connected to each other. In the CS view, components are present at different locations and connected by a network. In the more general view, locations can be physical (e.g., a room, a container), logical (e.g., an address in computer memory), or abstract (e.g., the location where a semaphore exists).
- **Resource** — Resources exist at locations and can move between them according to the locations' connections. In the general view, they can represent units data (and information), physical entities (including people), and derived ideas.
- **Process** — Processes execute and manipulate resources as they do so.

These concepts can be used to build a representation of a system's structure and operation, but there is one more concept required: the environment in which the system operates.

- **Environment** — Environments capture the world outside of the system of interest and how the two interact.

This generalization provides concepts that can be used to model essentially any type of system, from physical to logical, or systems that incorporate both. We note also that these concepts are scale-free — they can be used at any level of abstraction or representation. However, we have not actually defined what it means to *build* a model using this distributed systems approach. This, too, is very flexible. Models can be largely conceptual, and use the ideas of location, resource, and process as a means to help think about the structure and behaviour of a system. Or distributed systems models can be mathematical, as we will show in the next section. Finally, this metaphor can be used to build executable models, in the spirit of Birtwistle's Demos (15), where a programmatic description of the system (in terms of locations, resources, and processes) is run to simulate the behaviour of the system (17; 27; 26; 38). An early implementation of these ideas, Gnosis (38), has been used in significant commercial applications (9; 12; 13) derived from an industry-based research project (66).

The ability to compose models is important for modelling larger systems and ecosystems. During the modelling process, these systems can be decomposed into smaller parts, which can be modelled separately and then recombined, and which helps manage complexity.

To understand what composition look like in the distributed systems approach, we start by looking at the concept of interfaces. These define, for a model, the locations, resources, and processes involved in a composition. For a composition of two models to be valid, the interfaces in both models must match. Figure 1 depicts three models which compose together. When models with interfaces are not composed, the environment generates the events expected by the interface; when composed, the environment is replaced by a model. Also shown is an example of substitution: `Model C` can be substituted for `Model B` as the interfaces of the two models match; this allows a modeller to refine or increase detail in parts of a larger model.

Interfaces and composition seem to support the concept of local reasoning naturally. Obtaining such an account of reasoning requires a mathematical conception of the distributed systems metaphor on top of which interfaces and composition can be defined. Milner (100; 101) considers the concept of interface from the point of view of a quite abstract graphical theory of processes. Our notion is more directly grounded in the concept of a distributed system, but we conjecture that the approaches can be understood comparatively. Our approach is more directly concerned with the logical concept of local reasoning.

## Mathematical Theory

We begin by giving a formal framework for capturing the distributed systems metaphor that we are proposing as a basis for a semantically and logically well founded framework for modelling ecosystems of systems in the absence of locations. The basic theory of processes and their associated logics is technically essentially determined by the interaction between processes and resources, with locations playing a significant conceptual rôle only when the model-engineering concepts of interface, substitution, and local reasoning are considered, which we do in the following paragraphs. The results presented in this section for states $R, E$ extend to states $L, R, E$ (38).

### *Processes and Resources*

Our starting points are Milner's synchronous calculus of communicating systems, SCCS (98) — perhaps the most basic of process calculi, the collection of which includes also CCS (97), CSP (67), Meije (126), and their derivatives, as well as the $\pi$-calculus (99), bigraphs (100) and their derivatives — and the resource semantics of bunched logic (110; 119; 57; 117). The key components for our purposes are the following:

- A monoid of actions, $\mathsf{Act}$, with a composition $ab$ of elements $a$ and $b$ and unit 1;
- The following grammar of process terms, $E$, where $a \in \mathsf{Act}$ and $X$ denotes a process variable:

$$E ::= 0 \mid a \mid a : E \mid \sum_{i \in I} E_i \mid E \times E \mid \dots$$

Most of the cases here, such as 0, action, action prefix, sum, concurrent product, and recursion, will seem quite familiar.

Mathematically, this notion of resource — which covers examples such as space, memory, and money — is based on (ordered, partial, commutative) monoids (e.g., the non-negative integers with zero, addition, and less-than-or-equals):

- each type of resource is based on a basic set of resource elements,
- resource elements can be combined, and
- resource elements can be compared.

Formally, we consider pre-ordered, partial commutative monoids of resources, $(\mathbf{R}, \circ, e, \sqsubseteq)$, where $\mathbf{R}$ is the carrier set of resource elements, $\circ$ is a partial monoid composition, with unit $e$, and $\sqsubseteq$ is a pre-order on $\mathbf{R}$. The basic idea is that resources, $R$, and processes, $E$, co-evolve,

$$R, E \xrightarrow{a} R', E',$$

according to the specification of a partial 'modification function', $\mu : (a, R) \mapsto R'$, that determines how an action $a$ evolves $E$ to $E'$ and $R$ to $R'$.

The base case of the operational semantics, presented in Plotkin's SOS style (116), is given by action prefix and concurrent composition, $\times$, exploits the monoid composition, $\circ$, on resources:

$$\frac{}{R, a : E \xrightarrow{a} R, E'} \quad \mu(a, R) = R' \qquad \frac{R, E \xrightarrow{a} R', E' \quad S, F \xrightarrow{b} S', F'}{R \circ S, E \times F \xrightarrow{ab} R' \circ S', E' \times F'}.$$

This (rather general (98; 126)) notion of composition at the level of process does not explain the engineering concept of the composition of models, with its requisite notions of interface and substitution, that we discuss in the sequel.

Sums, which represented choices, recursion, and other combinators are defined in similar ways (6; 37; 38; 40).

A modification function is required to satisfy some basic coherence conditions (in certain circumstances, additional structure may be required (6)): for all actions $a$ and $b$ and all resources $R$ and $S$, and where $\simeq$ is Kleene equality,

- $\mu(1, R) = R$, where 1 is the unit action, and
- if $\mu(a, R)$, $\mu(b, S)$, and $R \circ S$ are defined, then $\mu(ab, R \circ S) \simeq \mu(a, R) \circ \mu(b, S)$.

This function specifies the *signature* of the model: the idea is to specify the modification function for the basic or *atomic* actions of the model, with more complex actions being generated by the coherent composition defined above (38; 6).

Given this set up, equality of processes is given by *bisimulation*, as defined in (39; 38; 6), which is written as $L, R, E \sim L', R', E'$.

### *A Modal Logic*

Along with a process algebra specify in this way, we can define a modal logic of its states, with a judgment of the form

$$L, R, E \models \phi$$

which is read as 'the property $\phi$ holds of the state $L, R, E$'. (Here we assume the states are 'closed' with respect to bisimulation; see (39; 38; 6) for the details of this, which we elide here.) As above, locations $L$ can be suppressed in the development of this idea.

This logical judgement supports the definition of the usual classical connectives, such as

$$\begin{array}{rcl} L,R,E \models \phi_1 \wedge \phi_2 & \text{iff} & L,R,E \models \phi_1 \text{ and } L,R,E \models \phi_2 \\ L,R,E \models \phi_1 \vee \phi_2 & \text{iff} & L,R,E \models \phi_1 \text{ or } L,R,E \models \phi_2 \\ L,R,E \models \phi \to \psi & \text{iff} & L,R,E \models \phi_1 \text{ implies } L,R,E \models \phi_2 \end{array}$$

and also the usual *action modalities*, which connect the logical assertions to the evolution of states, as

$$\begin{array}{rcl} L,R,E \models \langle a \rangle \, \phi & \text{iff} & \text{for all } a \text{ such that } L,R,E \xrightarrow{a} L',R',E', \\ & & L',R',E' \models \phi \end{array}$$

It also supports multiplicative connectives of the kind found in bunched logic (see, for example, (118)). Assume that there is also a monoidal operation $\bullet$ on locations. Then we get

$$\begin{array}{rcl} L,R,E \models \phi_1 * \phi_2 & \text{iff} & \text{there are states } L_i,R_i,E_i, \text{ for } i=1,2, \text{ such that} \\ & & L = L_1 \bullet L_2,\ R = R_1 \circ R_2, \text{ and } E \sim E_1 \times E_2 \\ & & L_1,R_1,E_1 \models \phi_1 \text{ and } L_2,R_2,E_2 \models \phi_2 \\ L,R,E \models \phi \mathrel{-\!\!*} \psi & \text{iff} & \text{for all } L',R',E' \text{ such that } L',R',E' \models \phi, \\ & & L \bullet L', R \circ R', E \times E' \models \psi \end{array}$$

The multiplicative connectives support the idea of reasoning about 'separated' parts of the model — in the sense of Separation Logic (121) — and local reasoning, which is directly relevant to our notion of *interfaces* between models. This idea is discussed below.

Finally, we remark that the connection between the location-resource-process states and the modal logic is described by theorems that establish, subject to a degree of care, the correspondence of bisimulation equivalence and logical equivalence: roughly speaking, two states $S$ and $T$ are bisimulation equivalent just in case they satisfy the same logical formulae. These results are elided here, with details being available in (39; 38; 6), with a slight mis-statement in (29).

### *Interfaces and Local Reasoning*

We have discussed, as illustrated in Figures 1 and 2, how our models are composed using interfaces (27; 28; 29; 10).

The multiplicative conjunction, $*$, provides the basis for an account of local reasoning about the composition of models through interfaces. Consider again Figure 1. We can identify here a local reasoning principle, or *frame rule* (75; 109; 139; 108; 117). We begin by setting up some notation for the states of the various component models depicted in Figure 3. The details of this set up may be found in (26). Here, we denote the composition of two models $M_1$ and $M_2$, via interfaces $I_1$ and $I_2$, respectively, as

$$M_1 \, {}_{I_1}|_{I_2} \, M_2$$

Then, we can make the following assumptions:

- let the model $M = M_1 \, {}_{I_1}|_{I_2} M_2$ have state $\mathcal{S}$;
- let the component models (of the composition of interest) $M_i$ have states $\mathcal{S}_i$, respectively;
- let the submodels $N_i$ have states $\mathcal{U}_i$, respectively; and
- let the interfaces $I_i$ have states $\mathcal{I}_i$, respectively.

Now we assume the following, writing $\circ$ for composition of states, for $i = 1, 2$:

- $\mathcal{S}_i \sim \mathcal{U}_i \circ \mathcal{I}_i$, $\mathcal{S} \xrightarrow{a} \mathcal{T}$, and $\mathcal{I}_i \xrightarrow{a} \mathcal{J}_i$
- $a \# N_i \backslash I_i$; that is, that the action $a$ is 'separated from' (denoted #) that part of the model $N_i$ that is not coincident with the interface $I_i$ (denoted $N_i \backslash I_i$) in that the execution of $a$ does not affect $N_i$.

Now, suppose that $\mathcal{U}_i \models \phi_i$, for $i = 1, 2$, and $\mathcal{J}_i \models \psi_i$. for $i = 1, 2$. Then we have the following frame rule:

$$\frac{\mathcal{J}_1 \models \psi_1 \quad \mathcal{J}_2 \models \psi_2}{\mathcal{T} \models (\phi_1 * \psi_1) * (\psi_2 * \phi_2)} \quad \begin{array}{l} \text{- } \mathcal{S} \xrightarrow{a} \mathcal{T} \text{ and } \mathcal{I}_i \xrightarrow{a} \mathcal{J}_i \\ \text{- } a \# N_i \backslash I_i \end{array}$$

This rule is sound with respect to bisimulation equivalence.

## Interpretation in Julia

As we have briefly touched upon at the beginning of this section, the basic concepts that form the distributed systems metaphor have been used to construct executable frameworks for practical model construction. The earliest implementation attempted was Dahl & Nygaard's Simula (42), which was an Algol simulation framework that mainly focused on the notion of processes. Further implementations such as Birtwistle's Demos (15; 16) or Gnosis (38) focused on extending the conceptual tool-set to notions such as resources or locations. In the context of this paper, we employ an implementation in the Julia language: the Julia `SysModels` package (17) is an improved, more modern implementation of these ideas that includes new capabilities such as composition of models.

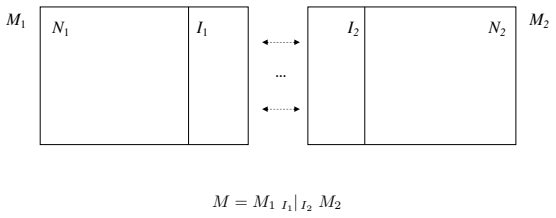



**Fig. 3.** Local Reasoning and the Frame Property

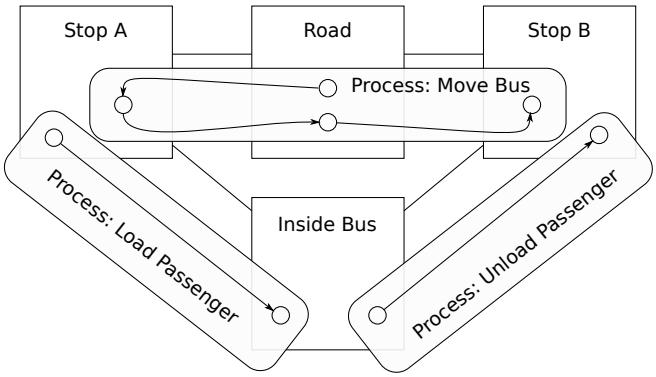



**Fig. 4.** Example model: passenger travelling on a bus. Squares represent location and the rectangles depict the processes moving resources between the locations.

The Julia Implementation

The `SysModels` package provides the constructs needed to build models — locations, resources, and processes — as well as the ability to execute them. We will use a simple example to demonstrate how models are written and executed, showing how the basic components are combined to form a representation of a system.

In this example, a bus moves between two stops and a passenger waits for the bus to arrive, boards the bus, and gets off the bus when it arrives at the next stop. Figure 4 depicts this scenario. The first thing we define in our model is the location structure:

```
using SysModels

loc_stopA  = Location("Stop A")
loc_stopB = Location("Stop B")
loc_road  = Location("Road")
loc_bus  = Location("Inside Bus")

link(loc_stopA, loc_road)
link(loc_stopB, loc_road)
link(loc_stopA, loc_bus)
link(loc_stopB, loc_bus)
```

Here, we have defined the four locations in this model: two bus stops, a road between them, and a location representing the inside of the bus. We then use `link` to define how these locations are connected, which determines how resources may be moved from one location to another.

After this, we need to define the resources that are used in the model. In this case, there are two: a passenger resource and a bus resource, which are defined using Julia structs. We also use `distrib` to set the starting locations of each of these resources:

```
struct Passenger <: Resource
end

struct Bus <: Resource
end

pax = Passenger()
bus = Bus()

distrib(pax, loc_stopA)
```

```
distrib(bus, loc_road)
```

Next we define the processes in this model. We start with the process responsible for moving the bus:

```
function move_bus(proc :: Process)
    @claim(proc, (loc_road, bus))
    move(proc, bus, loc_road, loc_stopA)
    release(proc, loc_stopA, bus)
    hold(proc, 2minutes)
    @claim(proc, (loc_stopA, bus))
    move(proc, bus, loc_stopA, loc_road)
    hold(proc, 5minutes)
    move(proc, bus, loc_road, loc_stopB)
    release(proc, loc_stopB, bus)
end
```

Processes are written as Julia functions. The `SysModels` package provides functions for controlling processes and manipulating resources:

- `@claim` — before a process can manipulate (move, remove) a resource, it must claim it. If the requested resources are not present, the process waits until they are; if multiple processes are trying to claim the same resource, they implicitly queue for it.
- `move` — after a process has claimed a resource, it can be moved from one location to another.
- `release` — when a process has finished with a resource, it releases it. Other processes may then claim it.
- `hold` — this pauses the process for a specified amount of simulation time.

In the code above, the process first claims the bus resource at the road location, before moving it to stop A and then releasing it. It waits 2 minutes, claims it again, moves the bus to the road, waits 5 minutes, moves the bus to stop B, and finally releases it.

The processes for loading and unloading the passenger are written in the same way:

```
function load_passenger(proc :: Process)
    @claim(proc, (loc_stopA, pax) && (loc_stopA, bus))
    move(proc, pax, loc_stopA, loc_bus)
    release(proc, loc_bus, pax)
    release(proc, loc_stopA, bus)
end

function unload_passenger(proc :: Process)
    @claim(proc, (loc_stopB, bus) && (loc_bus, bus))
    move(proc, pax, loc_bus, loc_stopB)
    release(proc, loc_stopB, pax)
    release(proc, loc_stopB, bus)
end
```

Here, the load process waits until it has claimed *both* the passenger and bus resources. It then moves the passenger to the Inside Bus location, and then releases both resources. Similarly, the unload process waits until it has claimed both resources, and then moves the passenger to Stop B. Claiming the passenger resource means the processes can move that resource into or out of the bus; claiming the bus resource causes the processes to wait until the bus resource—moved by the bus process above—arrives at the correct location.

Finally, we have to set up the model and run it:

```
model = Model()

proc_bus = Process("Move Bus", move_bus)
proc_load = Process("Load Passenger", load_passenger)
proc_unload = Process("Unload Passenger", unload_passenger)

add_startup_process(model, [proc_bus, proc_load, proc_unload])

sim = Simulation(model)
SysModels.start(sim)
SysModels.run(sim, 2hours)
```

This creates the model object, creates the three processes, and sets them to start when the model begins executing. Then it creates the simulation, which handles the execution of the model, and runs it for 2 hours of simulation time.

Although not used in this simple example, the Julia implementation supports composition of models. Models can define interfaces, which specify the locations and actions involved in composition. Then, two models can be composed:

```
composed_model = compose(model1, interfaceA, model2, interfaceB)
```

The `compose` function here returns a new model from the composition of `model1` (using its `interfaceA`) and `model2` (using its `interfaceB`).

The ransomware recovery model presented later in this paper is naturally much larger and more complex than the example given here. However, it is still constructed in a similar manner, using locations, resources, and processes to represent the different elements of the systems being modelled, and composition to construct the whole model from smaller sub-models.

The Stochastic Environment

So far, we have not described how the stochastic environment, within which a model is situated, is represented and integrated with the structural definition of models that is described in theory in Section 3.1 and implementation as above.

Our modelling set-up makes the choice not to incorporate stochastic definitions in to the definition of the process algebra and its logic (as is done in, for example, PEPA (60)). Rather, the stochastic existence of actions is represented at the level of the implementation. That is, suppose an action $a$ is defined in a model as in Section 3.1. In an implementation of that model, $a$ may 'fire' when a specified distribution for it is sampled, and the model executes according to the execution of a scheduler.

While this approach simplifies the definition of the semantics, as sketched in Section 3.1, it necessitates the giving of the an interpretation of the scheduler in the underlying semantics. This is done, for Gnosis, in (37; 38). The analysis for the Julia packages would be essentially the same.

## Practicalities

We have only sketched how the stochastic aspects of our models integrate with the approach above at the theoretical level, since the primary focus of this paper is not to repeat the description of a general modelling theory, but rather to experiment with such a theory and modelling framework in a security context. Therefore, we now describe how the stochastic aspects of our models are used in practice. This is closely related to the approach of Demos (15) and Tofts (46).

In elementary terms, executable models constructed using the above described approach behave like a structured Monte Carlo simulation (48; 142). Stochastic processes are employed as means of dealing with uncertainty, but at the same time maintain the generic aspect of the model. For example, in our specific security case, we are interested in the process of ransomware infecting devices. This might be influenced by the security posture of the organization at the level of both employee training and awareness, the technical defence mechanisms present, the ransomware strain involved including the entry point of the attack, the industry in which the organization that owns the devices operates, etc. However, not all these aspects are modelled explicitly: from the perspective of the ransomware and given our model goal, the relevant aspects are the infection probability, the attack duration and the ransomware behaviour — in our case the spreading mechanism. By experimenting with numerous combinations of parameters we get to unpack the conceptual complexity of the features: a very high infection probability, a small attack duration and a fast spreading behaviour can be used to represent a worm-like ransomware that is using a zero-day exploit, while a high infection probability longer attack duration and no spreading behaviour could mean a phishing attack on an organization with poor security posture. This represents an implicit stochastic aspect of simulation models.

Furthermore, stochastic processes are being used explicitly to describe behaviour by the means of sampling from probability distributions. This is generally being used in the case of complex or uncertain behaviors. For example, ransomware spreading behaviour can be represented using two processes: one that chooses attack targets and another that determines the time of attacking the targets. To describe a slow paced phishing attack that does not employ particular internal knowledge about the organization, we can use a uniform distribution for choosing targets at a regular, constant incremented time. Combined with the parameter describing infection probability, this already offers us the ability to represent slow-paced phishing attacks, but also fast-paced worm-like spreading ransomware in organizations with varying security postures.

Nevertheless, a lengthier discussion about parameter choices and what they represent at the model level will be presented in Section 6, with the complete parameter list being shown in Tables 7 and 8, in Appendix D. On a final note, this approach is computationally intensive. The experimental space, with a large number of parameter combinations, requires a lot of computational resource—in this case, a computing cluster—to execute in a reasonable amount of time. This is important, particularly since the number of model iterations and the size of the parameter space greatly influence the believability and representation power of the model.

## The Modelling Process

In the previous section, we presented the theory and implementation that supports the construction of models, along with a simple example to illustrate the approach. However, the process of modelling is more than just writing down the model: it is essential to know what aspects of the system need to be captured, and whether or not a given model is a suitable representation of the system of interest for its intended purpose. In this section, we look at the process of modelling as used in the construction of the ransomware model.

Given the nature of the recovery problem under ransomware and the complexities of an organizational environment — which more often than not lead to unexpected generative behaviour —, there is no surprise that a robust, modular and flexible methodology is required to construct a meaningful model. Given the information security orientation, such a methodology should facilitate rigorous reasoning about the design and behaviour of the modelled system. Because of this, our chosen approach is simulation modelling, using a distributed systems metaphor as described in Section 3 and underlined by a co-design process which will be described further. Traditionally, mathematical models have been constructed using a multi-stage iterative process named 'the classical mathematical modelling cycle' (88) (28). As it can be observed in Figure 5, this is comprised of six different stages:

- Constructing a conceptual representation of the phenomenon under study based on observing its domain. In our case, this would mean determining the relevant aspects of ransomware, organizations and recovery mechanisms that provide us with enough information about the recovery problem in this setting as to construct our model.

- Abstracting a candidate model based on the observations, using induction. The candidate model is based on the distributed systems metaphor to provide modularity, customizability and the ability to reason locally about its components.

- Deducing the mathematical consequences of the model. Succinctly, this means interacting with the model to produce a list of abstract properties or consequences relevant to the understanding of the system.

- Translating the mathematical consequences of the model by interpretation in the domain side. In our case, it follows that abstract consequences of the candidate model should be reflected back to the domain side in an explicit form. For example, a model consequence could describe the input parameter admin_nr as being an influential component of the output variable recovery_time. By interpretation, this would be translated in the domain as the number of admins corresponding to a help-desk in an office being an important factor in determining the recovery time of devices in that office.

- Validate the correspondence between the interpreted consequences in the domain side and the actual observed reality of the domain. Continuing with our example, this would mean confronting the relationship between the number of admins and the recovery time of devices with the observations about the domain in an attempt to either confirm or deny it.

- Update the conceptual representation of the phenomenon, based on the previous step.

These stages are repeated until a criterion of adequacy for the intended purpose of the model, often determined by the judgement of the modellers, is passed and the model is considered to be a good enough representation of the system under study. For additional information about this criterion in the system dynamics simulation literature, see Forrester (55). Nonetheless, the efficacy of this modelling process depends on certain key, usually unstated, assumptions about the modelling task:

- The structure and behaviour of the domain is clearly understood in conceptual or engineering terms. For example, the increase of device recovery time given an increase in device recovery requests is a well known phenomenon attributed to the inability of a network to transmit requests with the same speed after a certain congestion level has been reached. Such a phenomenon is a good candidate for mathematical modelling, perhaps in the context of testing network upgrades or extensions.

- The data that can be collected about the domain is essentially unambiguously identified. For example, modelling an organization's employees productivity levels for the sake of improving them might prove an extremely complicated task for mathematical modelling since, essentially, employees might be motivated by extremely subjective concerns and the interpretation of those concerns might differ from person to person.

- The questions that the model is intended to address are identified independently of the detailed design choices required for the construction of a model. For example, building a model for the purpose of optimizing the production time of hardware components in a fully automated manufacturing environment is well suited for the traditional modelling methodology described above. Contrarily, simulating the same system for the purpose of understanding its behaviour and only then deciding what can be optimized would be more suited to a different approach.

However, such assumptions are not compatible with real-world complex systems simply because the underlying components of those systems differ in their nature, providing the modeller with much richer challenges in terms of the design, construction and interpretation. Firstly, in our case, the structure of a communication network and of ransomware are understood in engineering terms, the structure of organizations can be considered as understood in conceptual terms, and yet, the employee behavior that severely influences the infection rate of devices or the possible behaviour of new ransomware strains is not. Furthermore, information

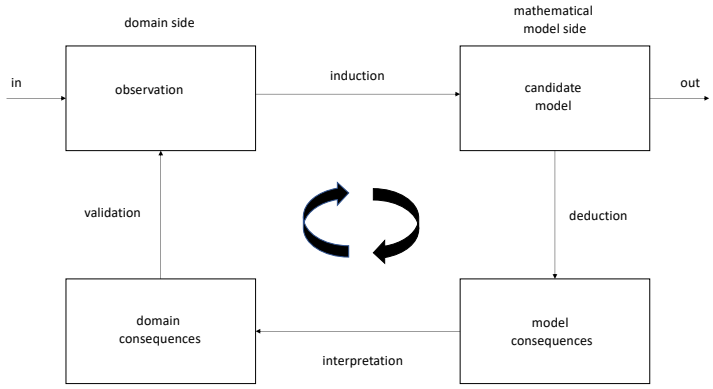



**Fig. 5.** The classical mathematical modelling cycle (28)

on employee behaviour might either not be available, hard to interpret or bound to various interpretations. Lastly, and we will expand on this particular point in the following paragraph, the separation between the questions the model should answer and the domain influenced design choices might not be desirable.

As we can easily observe with only the above examples, the traditional mathematical modelling approach is suited for phenomena that already have a significant theoretical understanding behind them, and that is particularly why the third assumption can be considered as legitimate in such cases. However, if the domain is still an active area of research, we argue that the knowledge about the domain and the questions the model can provide answers for are strongly related, and should be updated while iterating through the modelling cycle. In a sense, the knowledge about the domain provides an array of possible questions that the model could answer, but choosing a specific set of questions limits then the necessary information from the domain required for answering. This observation has led us to a methodological development called the 'Co-design modelling cycle'(28; 25), which can be observed in Figure 6 and is based on the explicit concept of a 'translation zone' that mediates the dual evolution of domain knowledge and model focus.

As defined in (25), 'Model co-design is a process that engages modellers and system stakeholders cooperatively in the acts of objective identification and model specification, design and construction with the aims of aligning model objectives with the needs of the stakeholders, and designing a model that is feasible given the limits of data availability, which are discovered as parts of the process.'. The essential difference between our approach to co-design and participatory design (44) or participatory modelling (132) is that the stakeholders and other expert knowledge participants do not provide expertise only at certain stages, but rather contribute to the whole process — in our case, the stakeholders were involved in all the modelling stages, from providing relevant information about the domain, to contributing to choosing the model structure, interpreting model results and even participating in code debugging sessions. The insights provided by the stakeholders directly contribute to the model, making this method closer to agile software development (1) than to participatory modelling. Similarly to the classical mathematical modelling cycle shown in Fig 5, the co-design cycle from Fig 6 does not have explicit termination and accuracy criteria. In both cases, the criteria must be determined case-by-case — and in the case of co-design, via co-creation in the translation zone —, but always respecting a few key general considerations relative to which the notion of accuracy must be calibrated: remembering that 'the map is not the territory' (82); appropriate level of detail; timeliness; and cost-effectiveness.

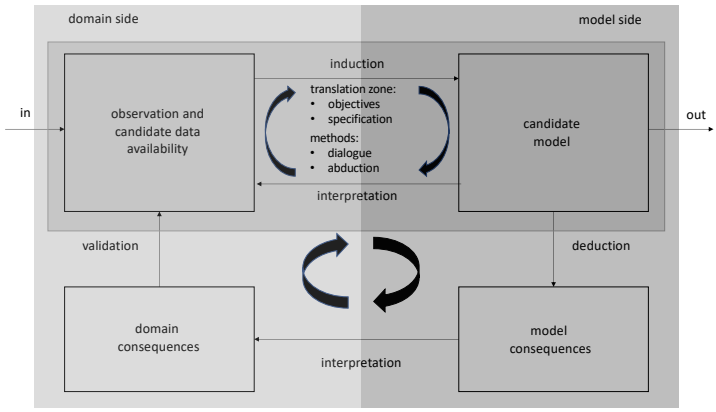



**Fig. 6.** Co-design in the translation zone (28)

## The Recovery Model

Having described both the theoretical background in Section 3 and the methodological aspects above, it is now time to focus on the actual model. Since the main goal of this model is to help organizations in deciding what allocation of recovery technologies best suits them based on an understanding of the consequences involved, we concentrate on a partially-generic model that allows for further customization of both the structure, because of the modularity provided by the underlying distributed systems metaphor, and parameters.

Making use of the compositionality provided by the method, we construct our organizational recovery model as a composition of four different sub-models: a device model, a network model, a server model and a malware model. The high-level architecture of this composed model can be observed in Fig. 8.

The **device model** is the most complex one of the four, containing information regarding the physical locations associated with the organization, abstract locations, types of devices and their associated recovery mechanisms, helpdesks' configuration and timings, employee movement patterns and expertise regarding recovery. A conceptual diagram of the processes present at the level of the device model is presented in Figure 7.

- **Resources** — devices, blank usbs, usb images, os images, recovery agent, image requests, image responses, network data, helpdesk admins

- **Locations** — work locations (home, offices, travel locations), network endpoint locations

- **Processes** — installation of images from usb, embedded, network-based or mixed, usb recovery, embedded recovery, network-based recovery, mixed recovery, fetching image from server, fetching recovery agent from server, admin delay, device movement

- **Interfaces** — network endpoints

Following the distributed systems metaphor, we conceptualize devices as resources and both the work and abstract locations as locations: work locations could include small or large offices, travel locations such as airports, coffee shops or the home; abstract locations exist mostly in the form of endpoint locations, an abstraction for the network switches or routers that connect devices to a network. As an implicit organizational policy, each large office is considered as hosting a help desk with a variable size that can help users perform a recovery process for their device if their level of expertise does not allow them to perform it on their own. The helpdesk employees are conceptualized as resources associated with the helpdesk. Furthermore, the employees are not being modelled explicitly, but rather as devices that move between locations and can perform work related activities or recover a device. The actual recovery actions performed by the devices or helpdesk employees are modelled as processes.

For example, when a device moves to a location, it obtains use of a network endpoint so it can send and receive data on the network by claiming one of these availability resources; when it leaves, it releases that availability resource so another device may use that endpoint later. Each device in the model has its own process. This process is responsible for all the device's behaviours, from movement, to recovery, to sending and receiving data on the network. As part of the configuration of the model, each device is set up with: a movement pattern (the sequence of physical locations to move to, and probability distributions determining the length of time it stays in each one) and a method of recovery to use. Furthermore, separate processes determine whether or not a

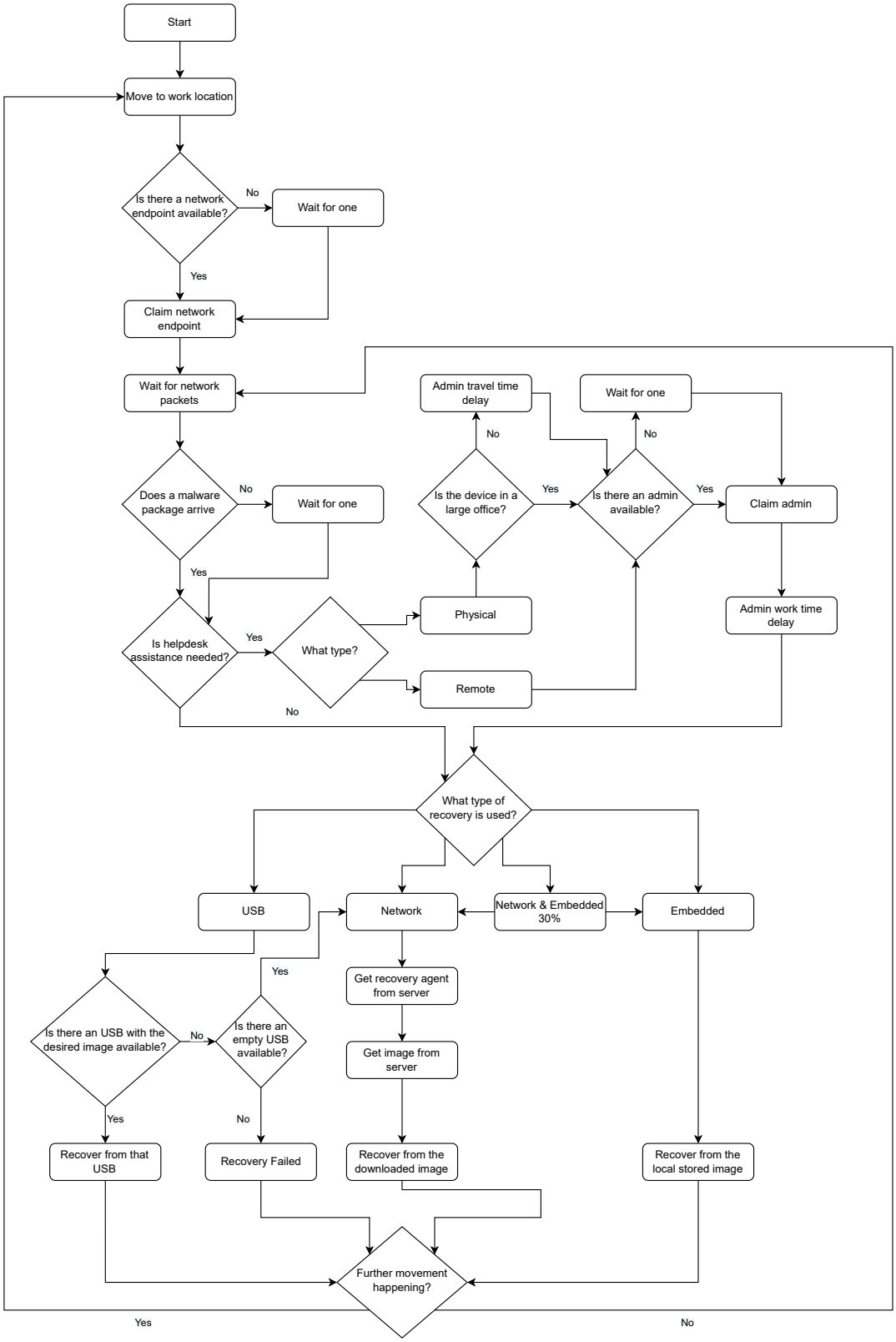



**Fig. 7.** Device Model Process Diagram. Rectangles show significant points in the behaviour of the process. Diamonds show where the process branches.

specific device can recover on its own and if not, what type of helpdesk assistance it requires. With these parameters, the device process executes. It moves the device resource from physical location to physical location according to the sequence, remaining in each one for a certain amount of time. If a particular device should recover, action indicated by the arrival of an infection package to the device, the device process initiates this.

As it was briefly explained in Section 2.2, in this paper, we look at four recovery methods.

- **USB Recovery** — A fresh OS is installed on devices from a USB stick. The device process tries to claim a USB stick resource with the OS image on it; if none are available, it tries to claim a blank USB; if no USBs of either type are available, and none *become* available, the recovery process fails. If a blank USB is obtained, the process must download the OS image by sending a request and waiting for the response, and writing it to the USB. This destroys a blank USB resource and creates a new USB stick resource with the image on it.

- **Network Recovery** — Devices request and receive an OS image over the network from an image server. The process starts by creating a request to download the recovery agent and moving it to the network endpoint so it can be sent to the server storage sub-model; it then waits for the response by claiming a response resource at the network endpoint. After receiving this, the process creates a request for the OS image, moves it to the endpoint, and waits for the response. For clarity purposes, we restate here that the network recovery process is behaviourally similar to, and uses the HP 'Sure Recover' (72) (Section 2.2.4) with network re-imaging as a primary reference point. This differs from PXE reimaging used in IT labs and for servers that would work on a LAN and not be available wherever the user is located.

- **Embedded Recovery** — Devices have a built-in storage capability that is used to hold an OS image for recovery. To model embedded recovery, the device process simply waits for the amount of time (as measured on real-world devices) it takes to restore from the embedded storage. This is based on the HP 'Sure Recover' (72) with embedded storage.

- **Mixed Recovery** — Combines the embedded and network-based recoveries. An allocation strategy is required. For example, devices in small offices could be allocated embedded recovery capabilities whereas those in big offices might rely on network recovery.

The above points, supported by Figure 7, describe the recovery choices and underlying actions that have been modelled. However, these do not encompass the stochastic, temporal nature of the model. Throughout all these steps in the process, there are time delays modelling the length of time it takes to, for example, verify an image after download, copy an image to disk, or run the installer. Additional delays are introduced if helpdesk assistance is required: helpdesks have a finite number of admins who can offer assistance remotely or in person — therefore, devices end up queuing for the helpdesk resource — and, if in-person assistance is needed, the admin might have to travel to a different location if the device in need is not located in the same large office as the admin.

The **network model** serves as primary abstraction for the communication network of the organization. It is formed of a series of abstract locations that represent network endpoints that a device can claim to connect to the network. A conceptual diagram of the processes present at the level of the network model is presented in Figure 18. The components of the sub-model are shown below:

- **Resources** — network data

- **Locations** — network endpoint locations, transit location

- **Processes** — transfer data

- **Interfaces** — network endpoints

Additionally, it includes a separate location representing data in transit. Structurally, all these locations are connected in the form of a graph of network segments representing the actual network routes packages would be routed through. This sub-model has one process, which claims resources that arrive at the endpoints, moves them to the transit location, and, after a delay suitable for the size of the data and the speed of the network segments it would traverse, moves them to the destination endpoint and releases them. If transfers are already ongoing when more resources are claimed or released, the process recalculates the time when the transfers will finish based on how throttled the network segments are.

The **server model** is the simplest one, with its components below:

- **Resources** — network data, os images, recovery agent, image requests, image responses

- **Locations** — server network endpoint location, storage location

- **Processes** — process messages, move network data to the endpoint

- **Interfaces** — network endpoint

At a high level, it is comprised of a network endpoint and a storage abstract locations. The storage location contains clean operating system images and the recovery agent necessary for the network based recovery. Behaviorally, a process awaits requests from the devices on the network endpoint and then delivers the operating system image requested back to the network model. A process diagram detailing the operations can be seen in Figure 20.

The **malware model** encapsulates the main aspects of the ransomware behavior. Structurally, it contains a single network endpoint location and three different processes: one that determines ransomware targets, one producing the timings when the infection packets are being injected into the network based on different probability distributions, and another one which determines if the targeted device actually gets infected and formats and sends the actual malicious packets. The process diagram can be seen in Figure 19.

- **Resources** — network data
- **Locations** — malware network endpoint location
- **Processes** — choose targets, determine timings, infect targets
- **Interfaces** — network endpoint

The combination of targeting, infection probability, timing distributions and duration of attack allows the modelling of ransomware behaviour as described in Section 2.1. Concerning the first process, we note that both specific location targeting and reinfection of devices is possible, even at the level of a single model iteration.

## Composed Model

As highlighted in Section 3, the main abstraction that practically facilitates the abilities to compose, substitute or local reason about the model using this approach is the interface. In the case of this model, the interfaces are defined at the network endpoints, essentially allowing the flow of network packets from a device to the storage server and backwards or from the malware endpoint to a device. The network model therefore becomes the glue that sticks the server model, device and malware models together. The server model composes with the network model at an endpoint; the malware model composes with the network model at an endpoint; the device model composes with the network model at *many* endpoints. After this, a request moved into the endpoint by the device model will be sent over the network by the network model, received by the server model, and the response sent back over the network model to the device model. However, such a request would only be transmitted if a malware packet was moved into the network endpoint by the malware model, routed by it to reach the endpoint of an actual device and the infection would be successful. The operation of an iteration of the composed recovery model would be comprised of the following steps:

Firstly, the four separated models would be initialized with their specific parameter set, which will be discussed in Section 6. This would construct the organizational structure and recovery policies of the modelled organization, define the capacity of the network, the available images for recovery, etc. At this moment, each model except the malware one has at least one process awaiting for network packets — devices awaiting a ransomware packet or a response from the storage server, the network process awaiting to route packets or the server awaiting requests — without knowing about the other models. The external processes that would bring the packets at the interface level are considered environmental processes.

Secondly, once the parametrization of the isolated models is completed, the composition of the models can be performed if pairs of interfaces exist at each model level. For example, in the case of the storage model composing with the network one, a similar interface object must exist in both of them. In the case of the network and device model, the same number of interface objects must exist in both. The composition of models actively transforms the environmental processes described above to internal model ones: if at the previous stage, a device would await a malware packet from the environment, now it would await it from the network model, but without knowing if other models were involved in the construction of the package along the way.

Thirdly, a simulation duration should be chosen, and then the execution of the model could commence. At the level of the device model, this would start the movement in between locations and the waiting for packets. Regarding the malware module, the complete list of devices and their distribution sampled timing of injection, including possible reinfections would be computed and then added to the network to be routed. The arrival of one malicious packet at the level of a device would trigger the associated recovery option on that device. Additional delays might happen here based on the user's ability to recover on its own and the need of helpdesk staff. Once the helpdesk interaction is determined an the timings applied, the actual recovery process between the device and storage server can be performed. We note here that the arrival of a malicious packet at device level during the recovery steps does not restart the recovery process: our conceptualization of both network and embedded enterprise recovery, as shown in Section 2.2.5, implies that recovery in a modern environment should be atomic. Reinfections can only occur after the recovery process has been successfully completed.

## Methodological Observations

Having described the model architecture, construction and operation in previous sections, we now focus on describing the implications of using the chosen methodological approach in our concrete setting. Starting with our conceptualization choice, we note that organizational modelling has proven to be a complex task even without considering recovery. This is because organizations themselves have been a difficult candidate for abstraction and, thus, multiple conflicting interpretations exist. For a comprehensive review of organizational metaphors, see (58). For a meta-classification of organizational metaphors, see (128). Nevertheless, we

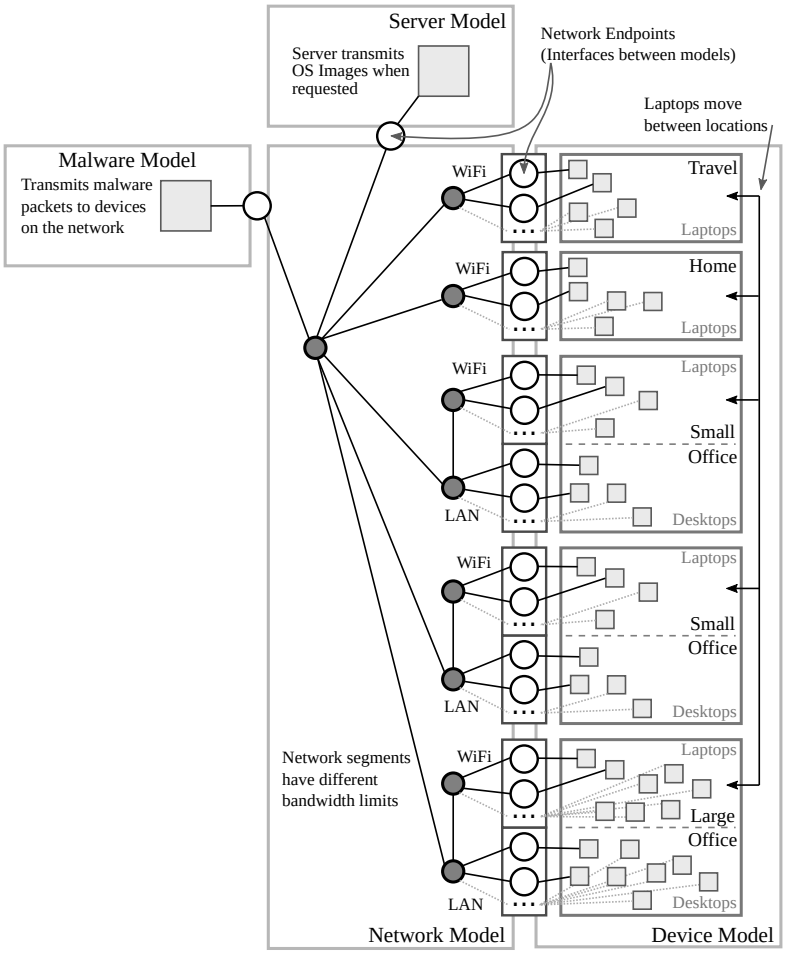



**Fig. 8.** Organizational Recovery Model

choose to think about organizations as being inherently compositional at the level of functionality, so we employ an interpretation based on different sub-components such as organizational goals, people, processes and technology (106), which construct an organizational boundary that effectively determines the organization.

In practice, information about these sub-components, or what could be considered the formal side of the organization, is provided by organizational structure and business processes documents, and further complemented with insights from stakeholders, expert knowledge, actual employees that take part in the modelled operations or KPIs. The knowledge co-creation process undertaken by these different parties — including modellers — in an attempt to construct a model representation that facilitates the achievement of the model goal is described in Section 4 and (25) and represents the co-design process.

Furthermore, it is important to note that our model is a partially generic prototype. Using the classification by Weisberg (135), we can view our model as 'modelling a generalized target'. The distinction is significant, since we are not producing a model for a precise client or company from an actual organizational structure, but rather attempt to derive a subset of elements relevant to an organization from a recovery perspective and then construct a semi-generic prototype of model that can be parametrized on a case by case basis. In Weisberg's terms, the target of our model is the subset of features relevant to recovery and common to all organizational instances at a certain level of abstraction, but of course limited by the project scope. Two criteria, necessarily but not sufficiently enough to ensure correctness must be satisfied in this set-up: firstly, to ensure that 'the relevant set of specific targets actually share the relevant features, such that an intersection of their sets of features is an informative generalized target'(135), and secondly, that 'a model can be constructed at the appropriate level of abstraction so that just those features can be modelled' (135). A longer discussion regarding the meaning of simulation correctness is outside the scope of this paper.

At an abstract level, the first criterion is tackled by the co-design process and the second one by the distributed systems metaphor and the theoretical considerations behind it. On one hand, the co-design process ensures us that the interaction of modellers, stakeholders and other experts produces a more suitable model representation for the model goal, precisely because the information about what constitutes relevant targets and features is not fully known by the modeller on its own and, multiple iterations of the design and construction cycle lead to a co-constructed ontology and representation of phenomena. On the other hand, the distributed systems metaphor can be used to construct models at essentially any level of abstraction that is decomposable into its basic notions of process, resource, location, interface and environment. In addition to that, the phenomena that can introduce

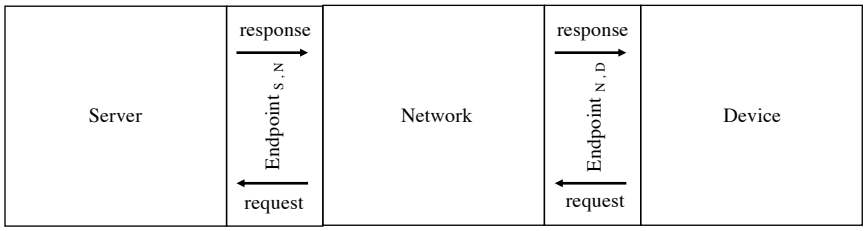



**Fig. 9.** A Simplified Recovery Model

uncertainty or do not have a generally accepted scientific knowledge base yet constructed are treated stochastically, as shown in Section 3.3.

In practice, we have chosen a high level of abstraction, focused on a small subset of relevant features and used the distributed systems metaphor described in (29) as underlying structural element. In this case, an organization is seen as a collection of locations where employees produce generic work over certain periods of time and can travel in between the locations. A communication network connects the locations and its characteristics such as speed, bandwidth or throttling factor have an impact on the work produced and timings for recovery. In other words, since the quality of work or its value for the company are highly subjective, we maintain the model focus on time, in the hope of using the timings to perform financial analysis when an actual concrete target is established and additional information is available. Furthermore, we justify our network based organizational conceptualization with two arguments. Firstly, the works Burns & Stalker(24), Mintzberg (103), Crainer (41), Eccles & Crane (47), Gulati et al. (64) or Baker (8) which are extremely relevant in the field of organizational theory all argue for different forms of network-based organizations, with Baker even arguing that 'the network form can be designed to handle product development tasks and market environments that demand flexibility and adaptability' (8). Secondly, given the focus on recovery and the fact that many technologies that are recovery-related are network-based, direct modelling at the network level was a natural choice because it offered the ability to translate from model consequences to domain consequences without having to unpack additional layers of conceptual complexity.

Moreover, the advantages of the methodology extend beyond the conceptualization of the system. The compositional approach to the design of these models facilitates the ability to reason locally about the underlying components, providing two primary advantages: modularity and the ability to focus the analysis on a singular model component without the need to reason about its relationships with other components. The modularity aspect complements the generality of the distributed systems metaphor and translates to scale-free modelling at any level of abstraction or representation.

To see how our approach to compositionality and local reasoning can be applied to such a setting, let's consider (following (29)) a stripped down, somewhat abstracted, version of the composite organizational recovery model. Here, for simplicity, we assume that composed models — Server–Network and Network–Device — have interfaces that are identical; that is, in terms of our definition in Section 3.1, this amounts to the interfaces from each of the models that are used in a composition being identical in each model. The simplified composite model is depicted in Figure 9 .

Now consider the composition of the device model and the network model. A device may request an image from the server by sending a request from the endpoint interface for transmission over the network to the server. The server's response, including the image, is transmitted over the network and received at the Endpoint interface, which now holds the image for receipt by the device:

$$\mathrm{Endpoint}_{N,D} \overset{response}{\longrightarrow} \mathrm{Endpoint}'_{N,D}$$

The availability of the image that is appropriate for the device can be expressed by a logical assertion such as

$$\mathrm{Endpoint}'_{N,D} \models \mathrm{Image}_X \wedge \mathrm{Device}_X$$

where $X$ denotes the required OS, so that $\mathrm{Image}_X$ denotes a proposition asserting that an $X$ image is available and $\mathrm{Device}_X$ denotes that the device requires the $X$ image.

Note that the separation condition, as defined above,

$$response \# \mathrm{Device}_X \backslash \mathrm{Endpoint}_{N,D}$$

holds. Consequently, applying the frame rule, we can substitute a different device model, $\mathrm{Device}'_X$, provided

$$\mathrm{Endpoint}'_{N,D'} \models \mathrm{Image}_X \wedge \mathrm{Device}'_X$$

can be verified.

This modularity of reasoning brings benefits similar to agile software design methods, such as reducing development time and increase focus, since both the stakeholders and modeller have a common way of understanding how the model evolves and can offer feedback or directions. Using an interpretation by Galison (56), the distributed systems metaphor can be considered to act as a 'pidgin language' between stakeholders and modellers, and improves the quality of the co-created knowledge. Additionally, given the security context, the ability to reason locally about sub-models at formal level increases the level of assurance the model provides while at the same time reducing reasoning time.

## The Experimental Space

After focusing on the explicit architecture and model building steps in the previous section, it is now the time to explore the experimental space that the model constructs and inherently operates in. Firstly, we describe the overall parameter space by taking into account the range of possible parameter values, their meaning and relevance with respect to the modelled phenomenon and, practical considerations related to the execution of the model over the parameter space. Secondly, we explain how meaningful organizational scenarios based on specific parameter choices can be constructed, exemplify a few such scenarios and justify those choices. Thirdly, we describe the validation procedures employed. Lastly, we present the results obtained from the execution of the model and discuss how real world organization can employ them to support security decision-making.

### Parameters & Scenarios

As previously explained in subsections 3.2 and 3.3, the presented organizational recovery model contains significant stochastic aspects and can be considered as behaving closely to a Monte Carlo simulation. Given the nature of the recovery problem and organizational environment, it is no surprise that such a simulation requires a relatively large number of parameters to describe and conceptualize recovery at a reasonable level of detail,.

Tables 7 and 8, in Appendix D, contain information regarding parameter names, type, value and meaning. Building on the stochastic aspects, the model employs two different types of parameters, fixed and variable. The fixed ones do not vary across simulation iterations and have a singular value related to the target organisational posture, whereas the variable ones do and, can either be represented as ranges of values or sets. Nevertheless, we firstly focus on the variable ones:

- *device_scenario* — represents the recovery technique that devices will attempt to use in the case of being hit by ransomware. As already discussed in Section 5, four types of recovery are available: USB recovery, full network recovery, embedded recovery and a combination of 30% embedded recovery at the level of laptops and the rest network based.

- *attack_scenario* — describes the part of ransomware behavior related to how the timings for the ransomware packages are being calculated. In more detail, five different probability distributions can be chosen from, for the timings sampling procedure: an uniform, exponential, F, uniform combined with exponential or uniform combined with the F distribution. By sampling from these distribution, we obtain the actual model time of a possible device infection. The shapes of these distributions are relevant in this context: the uniform distribution is used to represent attacks with a relatively stable infection rate — for example phishing attacks distributed by email at a slow rate to maintain a lower chance of detection —, the exponential and F distributions are used to describe the behaviour of fast spreading ransomware, with the main difference between them being that the F distribution increases slightly slower at the start and then decreases smoother — showing a more persistent attack that is harder to remove from systems — and the two combined distributions being used to showcase attacks that start in an uniform manner until they reach a certain infection threshold and then expand faster — such as email spam combined with internal spread capabilities. These behaviour examples are consistent with those described in Section 2. Furthermore, we note that the attack construction processes described at the level of the malware model in Section 5 are inherently compositional and additional ransomware behaviour can be introduced in the model by composing already implemented behaviour both at the level of a single malware model with multiple processes, or by composing multiple malware models at the level of the network.

- *infection_probability* — is being used to decide if a device targeted by a ransomware package actually gets infected or not. The value of this parameter can imply the simulated organization has a strong or weak security posture at the level of employee training or deployed countermeasures, or that the ransomware infection mechanism is rather novel or well-known. To explore the behaviour of the recovery techniques under attacks with varying success rates, we have chosen the following infection probabilities: 10%, 30%, 50%, 70% and 95%.

- *attack_duration* — represents the period of time that the ransomware attack takes to infect the targeted devices. We are using values of two, four and eight hours to describe attacks of different intensity. The number of attack packets used in a two and an eight hours attack is the same — given by the nr_of_samples parameter –, but distributing them during a smaller time period is bound to have a higher impact on the network.

- *admins_nr* — is used to determine the number of administrative staff deployed in a single help-desk. A help-desk is placed at the level of each big office location. Varying this parameter can imply different organizational policies about the number of available staff that can help users perform the recovery procedures if they are lacking the adequate skills to do so.

- *admins_need* — describes probability of a users' need of help to perform the recovery operations. We use different values in the range of [0.0, 1.0] in an attempt to illustrate varying skill levels at the level of employees. In case a user needs help, a time delay on the overall recovery process is introduced. This time delay varies with the type of help the employee needs: physical or remote. In the physical case, if the device is located in an office that has a help-desk, an admin resource can be claimed if available and the time delay specified by the parameter phisical_admin_time is applied. If the device does not have access to a help-desk in its current location, an additional time delay specified by the admin_movement_time parameter is also applied to account for the moving time of the admin. In the case of remote help being required, a single time delay is applied, specified by the admin_remote_time parameter.

However, if the variable parameters are being used to describe variations in the behaviour of ransomware, recovery techniques, organizational policies and staff training levels, then the fixed ones contain information about more stable aspects such as: the organizational structure, size of offices, number of devices, targets, movement patterns of devices, scaling factors for the time distributions, network speed, etc.

- *num_iterations & proc_num* — these two parameters are related to the parallel execution of the model. num_iterations determines how many times a set of parameters should be executed and proc_num shows how many different processes should be used. For example, in a case of forty iterations and four processes, each process will be used to execute ten iterations of the given parameter set.

- *nr_of_samples* — describes the actual number of ransomware packets used in the attack. In our simulation, we have opted for an attack size of three hundred packets, which is a value high enough so that all the devices have a chance to be hit and even reinfections occurring, but small enough so that the attack still resembles a ransomware infection and not a full scale, denial of service flood, given the size of the organization.

- *attack_targets* — contains the list of locations to be targeted by the ransomware packets. In the scenarios to be further presented, we have chosen to target all the possible locations where devices can be placed in.

- *physical_admin_time & admin_movement_time & admin_remote_time* — represent the additional time delays to be used in case an employee requires help with the recovery procedures. The values are being added over the actual timing of recovery, so if a user requires five minutes of additional help remotely and the actual recovery time for that device is 60 minutes, the overall recovery duration will be 65 minutes.

- *os_images* — contains the list of operating systems image resources available on the storage server. In the case of USB recovery, a windows10iso image is being used of 5GB. In the others, the devices first request a recovery_agent of size 350Mb and then an actual windows10wim file of size 4.7 GB.

- *max_office_devices & max_home_devices & max_coffee_devices & max_travel_devices* — represent the maximum number of devices that can be present in a type of location at any given time. For example, for a max_home_devices value of 30, no more than 30 users can work from home at the same time.

- *scale_uni & scale_dst* — are scaling factors used to ensure that sampling from the attack timing distributions do not yield results outside the desired limits given by the attack_duration parameter. For example, this ensures that sampling from the F distribution in the case of a two hours attack will not produce an attack timing of four hours.

- *num_office_desktops & num_office_laptops & num_small_office_desktops & num_small_office_laptops & num_travel_laptops* — represent the actual number of devices to be found in each defined location at the start of the simulation.

- *_speed* — in Table 8, Appendix D, parameters starting from server_speed and ending with travel_link_speed are being used to set the downloadupload characteristics of the network, based on the type of network endpoint used. Naturally, wired connections will be faster than wireless ones. Parameters containing 'link' in their description refer to the download or upload speeds present at the level of network endpoints.

- *office1_usb_images & office2_usb_images & small_office1_usb_images & small_office2_usb_images & home_usb_images & coffee_usb_images & travel_usb_images* — represent the available images already present on USB sticks in a certain location. If a device is using USB recovery and the location where the device is present does not contain USB sticks with that image pre-installed, additional time delays for downloading the image and flashing an USB stick, if available, will be introduced in a similar fashion to the network based recovery.

- *office1_usb_blanks & office2_usb_blanks & small_office1_usb_blanks & small_office2_usb_blanks & home_usb_blanks & coffee_usb_blanks & travel_usb_blanks* — are being used to set the number of available blank USBs for each location. In our scenario, these resources are only being distributed to offices, implying a certain organizational policy.

- *_movement* — in Table 8, Appendix E, parameters starting from travel_movement and ending with small_office_movement are being used to define the movement patterns of devices across locations. For example, in the case of office laptops, they can be placed in between zero and three hours at home — in the case of working from home for a limited amount of time — in between five and eight hours at the office and in between twenty minutes and two hours in a coffee shop. The order and precise duration of these movements is decided using Julia's shuffle function, which produces pseudo-random permutations of a given collection.

As it can be clearly seen from the above, the available array of parameters allows an actual organization willing to use this method to tailor its representation in detail and according to their own needs. This set-up allows the enterprise to compare different recovery options and parameters in a range of conditions and then combine this with other information about recovery, such as a security or financial analysis, to aid decision-making. When used in an actual real-world context, the parametrization of the model must be performed using both KPIs and employee knowledge, according to the principles of co-design. Nevertheless, to explore the descriptive power and usefulness of our model and method, we construct a basic enterprise scenario and detail the specific parameter choices from the above that cloud be considered relevant from the perspective of an organization.

We conceptualize our target organization as medium-sized, having an organizational structure comprised of two large offices, two small offices and additional adjacent locations such as a coffee shop, the home or travel locations such as airports,hotels, etc. The large offices host 40 employees, the small ones 20 employees, with each being equally split in using either laptop or desktop as work devices. A main difference between employees using desktops and laptops is that the ones using desktops are bound to their start location when working, whereas the laptop using ones are mobile and can move in between offices, the coffee shop, travel locations or home. Furthermore, the locations differ at the level of network connection speeds: we assumed that larger offices have better connections than smaller ones and that corporate locations such as the offices have better connections than general purpose locations such as a coffee shop or the home. The actual data about the network speeds can be consulted in Appendix E.

In addition to the different connection speeds, locations differ at the level of the available support resources. Each large office contains a help desk which provides users with help to perform recovery procedures. Nonetheless, support is not necessarily bound to large offices, since the administrative staff can also move to other locations or provide users with advice remotely — as explained in Section 5. In addition to that, we assume that large offices have access to USB sticks with pre-loaded operating system images, the small offices have access to USB sticks, but without images and the other locations do not have access to USBs.

In this organizational setting, we look at four different recovery techniques: USB, network based, embedded and a mix of network based and 30% embedded — as explained in Sections 5 & 6.1. It is worth noting that the model captures the active elements of the different approaches but does not represent the security of the different approaches. Here, we should consider the 'Sure Recover'-based mechanisms as secure in that they check the signature of the image, thus validating that the image is correct and as intended by the enterprise — for example, with the chosen enterprise AV systems installed. USB-based recovery has no validation of the image and either requires administrators who know BIOS admin passwords to initiate the installation or requires the computer to be kept in an insecure state — allowing USB boot or boot with no BIOS password. Furthermore, we looked at five different ransomware spread behaviours, by sampling attack timings from probabilistic distributions such as uniform, exponential, F and, combinations between uniform and exponential and uniform and F. Justifications for choosing these distributions can be found in Section 6.1, when describing the *attack_scenario* parameter. In addition to the temporal aspect, we test the network under difficult conditions that can resemble a fast-spreading ransomware by varying the *infection_probability* on a range of 15% to 95%, *attack_duration* between two, four or eight hours and, by setting the size of the attack, *nr_of_samples*, at three hundred packets.

In the following subsections, we present three different recovery scenarios in an attempt to illustrate how specific parameter choices can lead to meaningful organisational scenarios.

**Scenario 1: USB Recovery under uniform attack distribution**

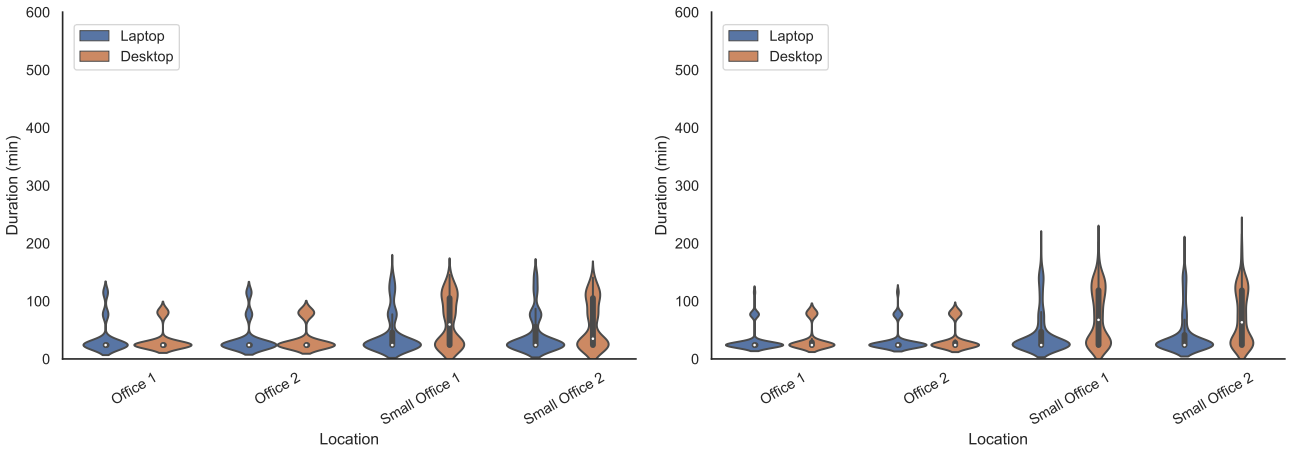
In the first recovery scenario we aim to determine the feasibility of using only USB recovery in the case of a temporally, uniformly distributed attack of varying severity, but a short time period of two hours. This can be interpreted as an intensive spear phishing attack that leads to ransomware being dropped on devices without an internal network spreading mechanism. Furthermore, we assume that users have the knowledge of performing an USB recovery process, so only one in thirty might require help-desk assistance, and that help-desks contain 10 administrative staff each. Therefore, the employees have a good knowledge of performing recovery and organizational policies ensures that the help-desks are well staffed. However, we must take into account that large offices have access to seven USBs pre-loaded with the right recovery image each, small offices have access to three empty USBs each and, the other locations do not have access to USBs. This implies a security policy of only using security vetted USBs, which are not available in travel locations.

Figure 10 shows the average recovery duration across locations, under a probability of infection of 30% and 95% respectively. These violin plots show the summary statistics and distributions of timings for devices across all the simulation runs (87). This variation in infection rate represents the level of preparedness the organization possesses in regard to phishing, both at the level of employee training and active countermeasures such as spam filters. As it can easily be observed, the large offices have a similar performance, regardless of the infection probability, with the highest values for desktops and laptops, close to 100 and 140 minutes respectively. This is due to two reasons: the uniform attack is unbiased at the level of targeting, so the ransomware packets do not flood a singular location and, the pre-loaded USBs do not require interaction with the network. Furthermore, we must note that in the 30% probability of infection case, an average of 58 devices recovered, whereas in the 90% case, there were 129.

However, in the case of small offices, the situation changes. For a 30% infection probability, both laptops and desktops have similar recovery timings of 186 and 176 minutes on the extreme end. Therefore, although they are using blank USBs which require network downloads, the severity of the attack is not enough to produce a greater delay. Nevertheless, the difference in number of devices, need of network interaction and, to a smaller extent even the need of physical help-desk assistance lead to a 33% recovery time increase in the case of laptops and a 76% increase in the case of desktops, on the extreme case. For the 95% infection probability, the recovery timing for laptops reach 225 minutes, whereas desktops go up to as much as 251 minutes. Interestingly, there is a small difference in between the performances of the two small offices that we attribute to the movement of laptops in between locations.

Therefore, this first scenario reveals some relevant facts for organizations that wish to employ USB based recovery. Firstly, even with a well equipped help-desk and users that know how to flash an USB, pre-loading the USBs with recovery images leads to

an almost twice as faster recovery rate in the case of a 95% successful spear phishing attack without automatic network spread. Secondly, under such conditions, a number of USBs representing almost a sixth of the number of devices (5.714) in the case of big offices is enough to prevent a steep increase in the duration of recovery. Thirdly, alterations to the security policy about the vetting of USBs must be made, because under the current one, all the devices in external locations — non-offices — failed the recovery process.




(a) Probability of infection 30%

(b) Probability of infection 95%

**Fig. 10.** USB Recovery under uniform attack distribution (Scenario 1)

**Scenario 2: Network Recovery under F attack distribution**

This recovery scenario depicts an organization with a weaker security posture than the above scenario, both at the level of user training — one in ten employees requiring admin assistance to perform device recovery — and at the level of help-desk staffing — each help-desk having 5 staff members. This organization is being targeted by a more virulent ransomware strain, modelled using the F distribution, but over a longer duration of four hours. This resembles the behaviour of a fast spreading ransomware, as presented in Section 2.1, particularly because when initialized with the values 50 and 8 as degrees of freedom, the probability density function of the F distribution increases steeply at the start and then decreases more gradually — in (77; 104; 137), the authors describe propagation graphs for worm-like spreading malware using similar types of distributions.

Figure 11 shows the average recovery duration across locations, again, under a probability of infection of 30% and 95% respectively. Compared with the previous scenario, we can observe differences in recovery time at the levels of desktops and laptops in both small and large offices. In the case of the 30% infection probability in large offices, the laptops recovery time varies between 50 and 98 minutes — with a median of 51 minutes —, whereas the desktops recover between 21 and 48 minutes — with a median of 23 minutes. In the case of 95% infection probability, laptops recover between 48 and 309 minutes — and a median of 71 minutes — and desktops between 23 and 300 minutes — with a median of 41 minutes. Furthermore, we must note that in the 30% probability of infection case, an average of 61 devices recovered, whereas in the 90% case, there were 141.

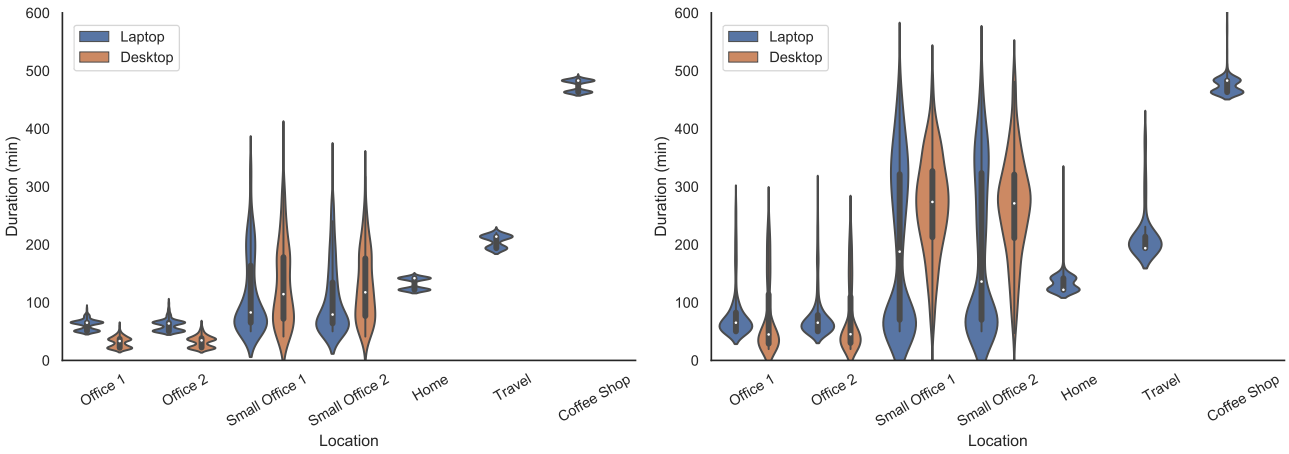
At the level of small offices, the increase in recovery time is steeper. For example, in the 30% case, laptops' recovery time varies between 54 and 385 minutes — with a median of 83 minutes — and, desktops' recovery time varies between 43 and 415 minutes — and a median of 125 minutes. In the 95% case, the laptops' recovery time varies between 53 and 588 minutes — with a median of 173 minutes — and the desktops recover between 23 and 557 minutes — with a median of 275 minutes. The median recovery timings for laptops at home are 150 and 125 minutes for the 30% and 95% infection probability. For the travel location, they are 225 and 200 minutes and, for the coffee shop, 487 and 486 minutes, respectively. However, since the attack duration is only 4 hours and the simulation time is 24 hours, a very small number of devices manage to both perform movement between locations and finish the recovery process.

A few observations can be drawn. Firstly, the big offices are more resilient to this type of attack, given the easier access to help-desks and the better link speed. The attack size of 300 network packets spread among 4 hours does not manage to produce a severe impact at the level of big offices in the 30% infection case. The 95% case is more interesting, with the median of desktops increasing with 56%, and the one for laptops with 71%.

Secondly, the desktops perform worse than laptops in the small office locations. Although peculiar at first sight, this represents an effect of laptops being able to move between locations, combined with the 24 hours simulation time: the number of laptops in small offices decreases because of movements, and those that are targeted early in the attack manage to complete the recovery, so the overall recovery time of successful laptops decreases.

Thirdly, the extreme values are significantly higher for the small offices. For example, even though on average, in the 95% case, a desktop manages to recover in around four and a half hours — which is less than the overall attack duration and means the device can get reinfected — some could take even more than nine hours, which is the equivalent of more than an entire day of

work. Because of this, organizations might seek alternative recovery processes, especially if a priority of recovery between employees exists.




(a) Probability of infection 30%

(b) Probability of infection 95%

**Fig. 11.** Network Recovery under F attack distribution (Scenario 2)

**Scenario 3: Mixed Recovery under Exponential attack distribution**

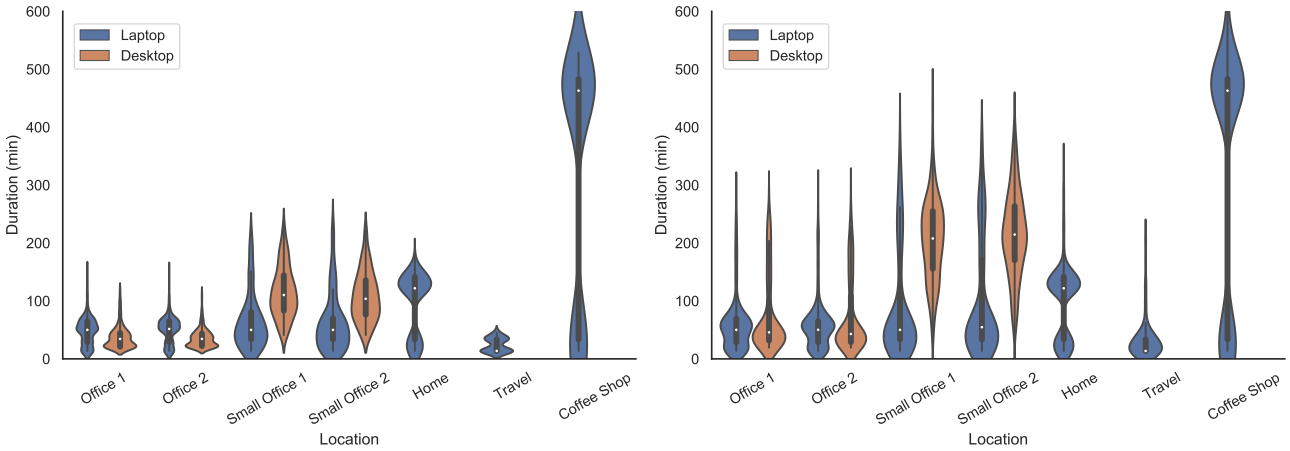
In this scenario, we maintain the same set of parameters for the organization — one in ten employees requiring admin assistance to perform device recovery and, each help-desk having five staff members — and the attack duration of four hours, but we change the recovery method and the attack distribution. We use an exponential distribution, which has a higher potential of increasing the network throttling because the distribution of ransomware packets is steep from the beginning, compared to the F distribution. However, the mixed recovery method used means that 30% of the laptops use embedded recovery in all the locations except the travel one. All the laptops in the travel location use embedded recovery.

Figure 12 shows the average recovery duration across locations, again, under a probability of infection of 30% and 95% respectively. We must note that in the 30% probability of infection case, an average of 63 devices recovered, whereas in the 90% case, there were 150. In the case of big offices, we observe that laptops' recovery time varies between 13 and 175 minutes — with a median of 49 minutes — in the 30% case and 20 and 325 minutes — with a median of 53 minutes — in the 95% case. The desktops' recovery time varies between 15 and 135 minutes — with a median of 30 minutes — in the 30% case and, between 25 and 328 minutes — with a median of 51 minutes — in the 95% case.

Furthermore, analysing the performance of small offices yields the following results. The laptops' recovery time varies between 13 and 265 minutes — with a median of 49 — for the 30% infection probability and 13 and 450 minutes — with a median of 54 — for the 95% infection probability. The desktops' recovery time varies between 38 and 265 minutes — with a median of 108 — for the 30% infection probability and, between 42 and 375 minutes — with a median of 210 — for the 95% infection probability.

In addition to that, we can easily observe that a significantly higher amount of laptops manage to complete the recovery process in a non-office environment. In the case of home, laptops manage to recover for both infection probability cases on average around 125 minutes. When considering the coffee shop, the median for laptop recovery is around 460 minutes for both infection probability cases. In the travel location, considering that the laptops use only embedded recovery, the median recovery time is 13 minutes.

This allows us to draw some relevant observations about this type of recovery techniques allocation. Firstly, using embedded recovery on 30% of the laptops has reduced the medians of device recovery in both in the 30% and 95% infection probability cases, regardless of device type, when compared to the previous scenario. Particularly in the case of small offices under 95% infection probability, the median of recovery time has decreased from 173 to 54 minutes in the case of laptops — a 68.78% decrease — and from 275 to 210 minutes in the case of desktops — a 23.63% decrease. This reveals an interesting insight for organizations: deploying a faster recovery method across a subset of all the devices in an organization can have a positive impact on all the devices if the organization was mainly relying on network-based recovery previously. Of course, this is not a 'silver-bullet' type of solution, because the extreme values for laptops and desktops are still 450 and 375 minutes in small offices, but the reduction at the level of medians shows that this type of approach is viable. Secondly, recovery in remote locations becomes more feasible, since more employees now have the chance to complete recovery before moving to another location — the coffee shop and home locations in Fig 12 show this clearly when compared with the same locations in Fig 11, because although the medians are similar, the number of points differs greatly. At the level of the travel location, even the medians differ because embedded recovery is used on all the laptops.




(a) Probability of infection 30%

(b) Probability of infection 95%

**Fig. 12.** Mixed Recovery under Exponential attack distribution (Scenario 3)

## Overall Behaviour and Model Sensitivity

The previous subsection was focused on providing detailed information about the model parameters and, exemplifying the model utility by showing three scenarios that could be useful for organizational decision-making with respect to security; we now focus on more general characteristics of the model.

Firstly, 9000 configurations of variable parameters — as explained in 6.1 and shown in Table 7 — have been used for the model execution. Each configuration has been run 50 times, so in total, the model execution is comprised of 450000 different runs. Each run employed both fixed and variable parameters as detailed in Section 6.1 and, elements such as when a ransomware packet is being sent on the network, to what target or to which location a device might move are being sampled for stochastic variables that represent the environment in which the model operates. The execution of the model was carried out in a distributed computing environment, across multiple cluster nodes with an 8GB memory limit. The size of the produced model output files was 4.42GB.

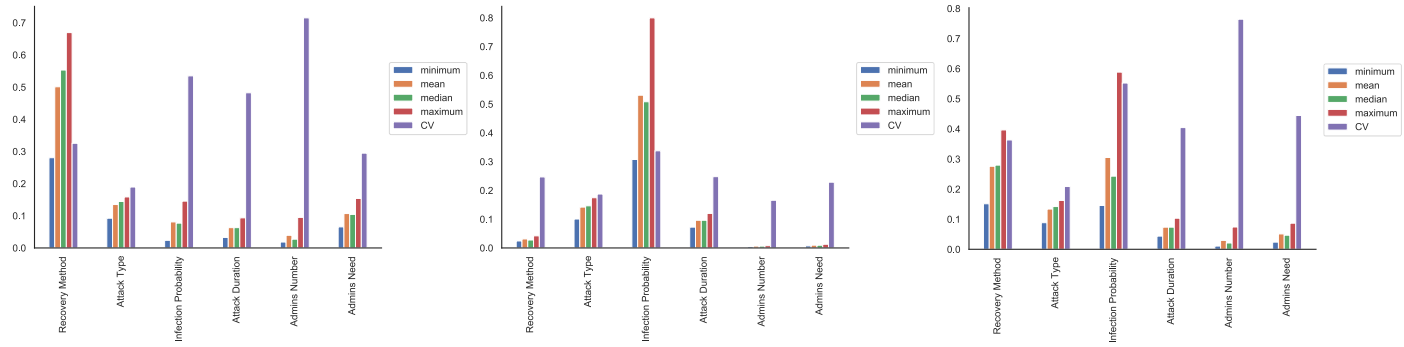
## PAWN Sensitivity Analysis: Method

Given the complexity of the phenomena under study and the stochastic nature of the modelling approach, we employ sensitivity analysis in an attempt to better understand the relationships between input and output variables and increase confidence in the overall model behaviour. We have chosen to use the PAWN sensitivity analysis method as described by Pianosi & Wagener (114), (115) and further developed by Baroni & Francke (11) for a series of reasons: allowing both categorical and numerical data to be used as input, offering information about the relevance of input variables with respect to output variables and at the same time describing the variability that changes in the input variables produces at the level of output, good performance with respect to multi-modal data — especially because of the ransomware behaviour mechanism, but also given the composition of multiple distributions.

In very brief terms, PAWN sensitivity analysis focuses on two metrics: the PAWN sensitivity index which describes the relevance of an input variable with respect to an output variable and, the coefficient of variation which is a measure of general variability produced during the variation of the input parameters. The key aspect of the method is the usage of cumulative distribution functions for output quantification instead of probability density functions or other traditional variance-based methods. The PAWN sensitivity index of an input variable with respect to an output variable is computed as the statistical difference between the unconditional CDF of the output variable — obtained by varying the input factors at the same time — and the conditional CDF of all the other input variables except the one under study. This measure is known as the Kolmogorov-Smirnov statistic (81) (127), a well-established method for calculating the distance between multiple cumulative distribution functions. The coefficient of variation is calculated as the division of standard deviation by mean, over the previously computed indexes. In layman's terms, a high sensitivity index implies that the specific input variable has a high impact on the value of the output variable, whereas a high value for the coefficient of variation translates to the input variable producing high amounts of variation on the output, but not necessarily producing severe output changes.

## PAWN Sensitivity Analysis: Results

We employ the above described method to analyse the relationships between the six variable input parameters described in Section 6.1 and three output variables: the average recovery duration, total recovery duration and the average number of devices recovering. The results are shown in Figure 13 below, and Figure 21, Figure 22, Figure 23, Table 2, Table 3 and Table 4 in Appendix C. The reason for choosing this specific output variable configuration is twofold: firstly, these three output variables represent relevant metrics for guiding organizational recovery on their own and, secondly, the behaviour shown with respect to the total recovery duration acts as a useful verification checkpoint for both the average recovery duration and the average number of devices recovering.




(a) Average recovery duration. (b) Average number of devices. (c) Total recovery duration.

**Fig. 13.** Sensitivity Analysis

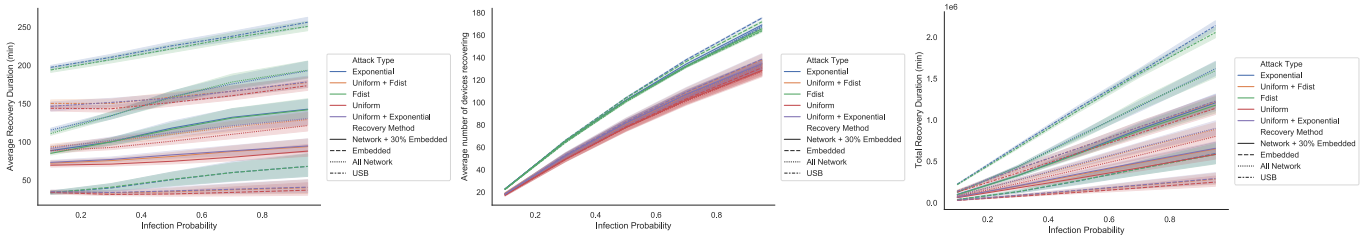
Starting with Figure 21 and Table 2, we can observe that the recovery method, type of attack and users' need for admins have the biggest impact on the average device duration of recovery. The extremely influential, maximum sensitivity index value of 0.669556 for the recovery method can be attributed to the fully embedded recovery mechanism and, reveals the fact that deployment of such a mechanism at the level of all devices in an organization can drastically reduce the impact of other factors on the average recovery time. With respect to variation, the highest cv scores were obtained for the number of admins, infection probability and attack duration. However, this shows a complete separation between the high-impact-producing input variables and the ones producing low-scale variations, which has led us to further analyse the total recovery duration.

In Figure 22 and Table 3, we observe a different variable allocation with respect to the average number of devices recovering. The infection probability, attack type and attack duration are the most influential and, the infection probability, attack duration and recovery method produce the highest amount of variation. Although one might expect a higher sensitivity index value for the recovery method, this is not the case since the individual simulation time of five days allows all infected devices to recover, in the end. Thus, the impacts of the recovery method, number of admins and users' need for admins are drastically reduced.

Now, since the total recovery duration time is dependent on both the recovery time of singular devices and on the actual number of devices that manage to completely perform the recovery procedure, we would expect that the sensitivity analysis for the total recovery duration time to maintain the trends from the analysis of average recovery duration and the total number of devices. Figure 23 and Table 4 reveal the following: the most impactful variables were the recovery method, infection probability and attack type and, the ones producing the most variation were the number of admins, infection probability and the users' need for admins. As it can easily be observed, the recovery method was the most influential input variable in 21, the infection probability in 22 and the attack type was the second most influential in both cases. The difference between the index values of the recovery method and infection probability was only 0.036667, so their impact can be regarded as similar. With respect to variation, the number of admins and infection probability maintained their leading trends from Figure 21 and 22. Therefore, our expectation was confirmed, since the sensitivity analysis of the total recovery duration did conserve the primary impact insights from above.

Although insightful about the general behaviour of the model, the sensitivity analysis above does not reveal enough information about the types of relationships between the variables, because it simply does not take into account their structure, but only the input/output changes. Consequently, we proceed to analyse Figure 24 to Figure 35 in an attempt to better understand how the structure of the input variables affects the value of the output ones.

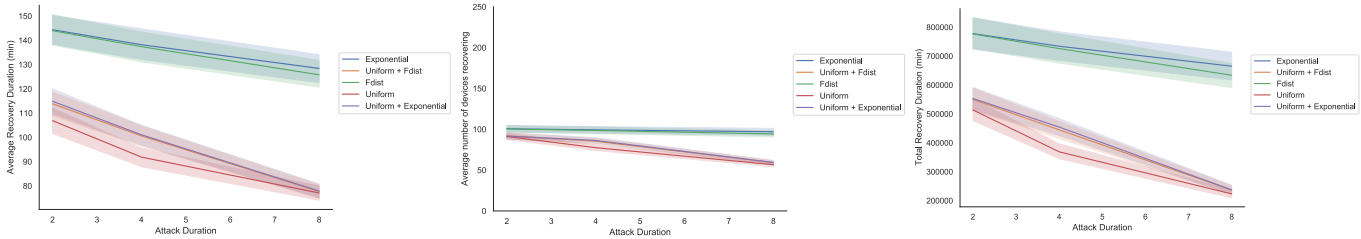
- **Infection Probability:** Figure 14, 24 below and 25, 26 in Appendix C are focused on the impacts of the infection probability, which was deemed the second most influential variable. A separation of classes of attacks based on the type of distribution used is immediately visible, but most clearly in Figure 25. A similar observation can be done about the recovery techniques. Given the stochastic aspects of the model, the produced output is surprisingly structured: for example, USB-only recovery under exponential types of attacks has the worst performance across all the diagrams and the full embedded recovery has the best performance regardless of the attack type. However, some additional insights can be gathered from the intermediary classes: in Figure 24, we can observe that at around 0.5 infection probability, the performance of USB-only recovery under uniform and uniform-exponential attack classes becomes better than the performance of network-only recovery under exponential types of attacks. Since in the real world, uniform-exponential attacks happen more often than purely exponential ones — the uniform part of the attack can be viewed as reconnaissance actions or an initial phishing campaign —, we can argue that for organizations with a low-security posture, USB recovery could be a viable option, especially if the only other one is purely network based. Furthermore, Figure 25 shows the benefits of a composite allocation of recovery techniques — 30% embedded recovery and 70% network-based: the distance between a full network approach and a composite one is clearly visible at the level of uniform attacks, but the separation zone between the distributions increases even more at the level of exponential attacks. In other words, the more virulent an attack is and the lower the security posture, the better will our composite approach behave when compared to a fully network-based one.
- **Attack Duration:** Figure 15 below and 27, 28 and 29 in Appendix C are concerned with the attack duration. As expected, since the total amount of network packets allocated to an attack is fixed, the more the attack takes, the less network throttling manifests, so the relationship is inversely proportional. Furthermore, because the individual simulation time is five days, almost all the devices manage to recover regardless of the attack type, which is exactly what can be observed in Figure 29. Interestingly,




(a) Average recovery duration. (b) Average number of devices. (c) Total recovery duration.

**Fig. 14.** Impact of Infection Probability on the total, average duration and the number of devices recovering.

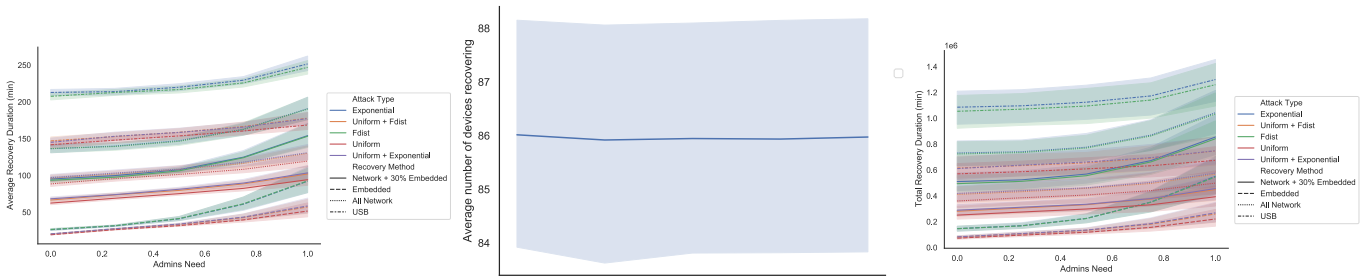
in such a case, more devices manage to recover in the case of exponential attacks because they get infected earlier and the throttling spread over five days is not influential enough, whereas, under an uniform attack, the last devices to get infected might not manage to complete recovery.




(a) Average recovery duration. (b) Average number of devices. (c) Total recovery duration.

**Fig. 15.** Impact of Attack Duration on the total, average duration and the number of devices recovering.

- **Need of Admins:** Figures 30, 31 and 32 focus on the users' need for admin help while performing the recovery procedures. Particularly in figures 30 and 31, we can observe that the attack type and recovery technique generate similar separation classes to Figures 24 and 25. This confirms the insights of the sensitivity analysis, which described the infection probability as a more influential input variable than the need for admins, especially since in Figure 31, the highest values of the need parameter lead to lower values of recovery time than in the case of infection probability. Similarly to Figure 29, Figure 32 depicts the low influence the input variable has on the total number of devices managing to completely recover, which is an effect of the five-day simulation time.

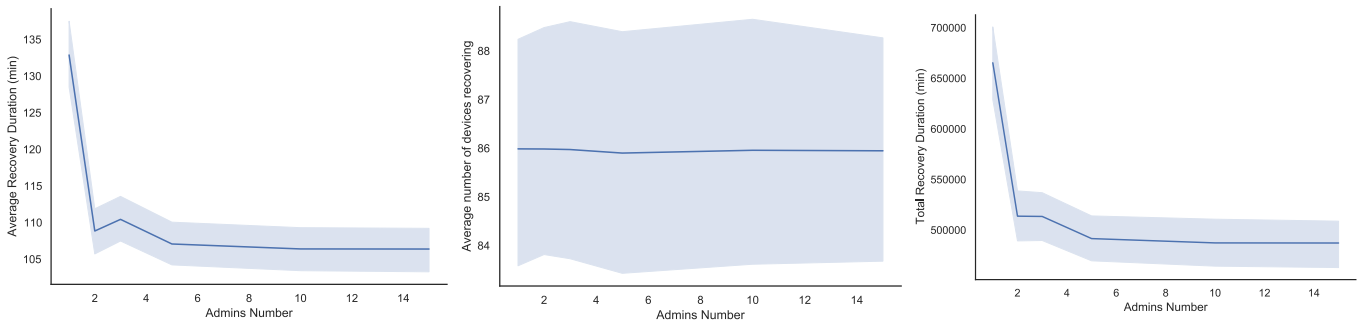



(a) Average recovery duration. (b) Average number of devices. (c) Total recovery duration.

**Fig. 16.** Impact of Admins Need on the total, average duration and the number of devices recovering.

- **Number of Admins:** Last but not least, Figures 33, 34 and 35 target the number of admins allocated to each help-desk location. As we can easily observe in Figures 33 and 34, given the model configuration, an optimal value for this parameter can be found in the interval [2, 5]. The small recovery duration increase that can be observed in Figure 33 around the value of three for the number of admins is a consequence of admin behaviour: if three admins are allocated to a help desk in a large office and then some other small offices are hit by ransomware, up to two admins could get moved there to assist with recovery procedures. This can lead to a situation where more admins end up being present in small offices rather than large ones, which can in turn increase the waiting time for users who require physical admin assistance. Furthermore, this particular result suggests a possible

need for improvement in admin workflow policies: if the model under analysis would have been built for an actual organization, this type of anomaly would reveal an extreme case where the admin policies would not work as previously expected.




(a) Average recovery duration.

(b) Average number of devices.

(c) Total recovery duration.

**Fig. 17.** Impact of Admin Number on the total, average duration and the number of devices recovering.

## Verification & Validation

Given our intention of showing that the above-described model represents an instantiation of a modelling methodology able to produce useful models for supporting security decision-making, we must now ensure that the model's conceptual reality is representative of the depicted scenarios. In the following lines, we describe a series of verification and validation procedures employed to do so.

- **Model conceptualization** — As a particularly relevant aspect of the Co-design cycle described in Section 3, we ensured a consensus was reached between a literature-only representation and expert knowledge. The ransomware, network, organization and recovery methods behaviour were constructed based on an extensive literature review for ransomware — which in turn led to a selection of features that included a targeting distribution, a timings distribution, infection probability and attack duration —, a well-known conceptual representation in the organizational literature for the organization — the network organization —, a simplified version of actual internet routing for the network and technical documentation for the recovery methods. Furthermore, additional improvements were brought in by expert knowledge, until a state of considered model utility was reached. As described in the Co-design methodology, this conceptualization was not fixed in an initial design phase but continued changing throughout development.

- **Antibugging** — Workflow validity checks were introduced at the level of the code to check whether quantitative or structural constraints were violated during execution. Examples could include the maximum number of users in a type of office, the users' movement patterns or the location they end up in, the minimum and maximum network speeds available or if ransomware packets are being transmitted by the ransomware model after the given attack duration.

- **Prototyping & Walkthrough Iterations** — The model construction process started with basic, compositional models for the devices, network, storage server and ransomware that were further developed and continuously validated by expert knowledge. To ensure this was possible, multiple walkthrough iterations were carried out for both separated and composed models. Furthermore, some runs made use of placeholders for an easier structural understanding, whereas others included stochastic input for the assessment of behaviour.

- **Real-World Data** — The recovery timings for the USB recovery method and the network speed constraints were manually gathered and then validated by expert knowledge — particularly the office distribution changes — and comparison with UK network speed statistics.

- **Event Tracing** — Based on the nature of the model, we were able to use a dual system of event tracing at the level of locations and timings. This helped with debugging once unexpected behaviour was manifested, but at the same time simplified analysing two different instances of the same model, in different stages of development. For example, a previous iteration of our recovery model included only an uniform attack spread of a constant rate. The current one allows for much greater customization by allowing different input distributions for spread and timings. Nevertheless, setting the right parameters for the current iteration with respect to rate and speed leads to similar behaviour to the previous iteration, therefore conserving consistency,

- **Sensitivity Analysis** — Sensitivity analysis was, again, used with a dual purpose. First of all, it confirmed that the choice of input parameters had an influence on the outputs — which was expected, but useful. Secondly, the general-level insights produced by it were used as verification checks for the examples and the structural analysis of the features. For example, the sensitivity analysis in Figure 23 reveals the infection probability as more influential to the total recovery duration than the attack duration. Figures 25 and 28 confirm it at the level of the Y-axis scale.

Nevertheless, we end this sub-section on validation with the idea that especially in an organizational context, the verification and validation procedures must be the result of an alignment between modellers, experts and stakeholders. For sure, traditional approaches such as antibugging and structured walkthroughs are still useful for ensuring the agreed-upon model conceptualization is implemented and the actual model iteration is understood. However, we must take into account that the organizational modelling goal is not only to produce an accurate representation of a phenomenon — accurate to what criteria could be a legitimate question — but an accurate representation useful for a practical outcome. This is precisely why stakeholders have an increased presence during model design and development through co-design: on one hand, they can introduce additional environment-specific insights that can act as conceptual optimizations for the model — the scope of an organizational model is smaller than the one of a purely scientific one — and on the other hand, they gain an increased understanding and belief in the usefulness of the model that can lead to more real-world outcomes.

### The Model as a Management Tool

As stated at the end of the previous paragraph, one of the innate goals of an organizational model is to produce some form of real-world outcomes either directly — if the model is being used without human intervention — or indirectly, by supporting decision making. Our focus is on the latter. To support such decision-making, the model must be parametrized and validated accordingly. However, since the presented model acts as a methodological example, we assume with the help of expert knowledge that the modelled organization acts as a reasonable surrogate for a real one. Given this, we describe in the following lines, a list of possible actions that decision-making factors might implement at the organizational level, based on the model.

Firstly, precise policy changes could be enacted at the level of help desks. For example, the policy regarding admin movement between offices could be altered such that no more than half the initial number of admins in one help desk is allowed to be out of the initial location at once. Furthermore, the number of admins in a help desk could be capped at a value in the interval [2, 5]: the model reveals this interval as producing the highest rate of reduction in total recovery time. A fixed value can then be decided upon based on additional hiring or economic policies that the organization might have interacting with its security posture.

Secondly, organizations can use the model to test different configurations of recovery techniques and then analyse their performance and benefits. For example, we have observed that the recovery techniques employed form classes of recovery behaviour that are reducible: on average, the organization displayed a similar recovery duration when under an uniform rate attack but using USB based recovery and, when under an exponential rate attack but using only network recovery. Based on historical data and risk appetite, organizations might consider one class of attacks as a priority, and therefore base their recovery technology choice on that. In our execution, a combination of 30% embedded recovery and 70% network-based recovery mechanisms on laptops was shown to produce a significant decrease in recovery time across different locations, particularly when compared with the more rigid but often used, complete network approach. This led to an increase in the overall number of devices that managed to perform recovery at all, even in locations where the embedded recovery was not deployed, due to a reduction in network throttling. Therefore, organizations might choose to experiment with the exact percentage value and the locations in which different recovery technologies are deployed. Possible deployment strategies could be focused on the criticality of the employee tasks to the organization, departmental budgets or organizational hierarchy.

Thirdly, the choice of inputs and their sensitivity with respect to the outputs can be used to highlight possible areas of further development, if their degree of control is being taken into account. In our case, although the attack type and duration were seen as having a somewhat medium impact on the recovery duration, they are not controllable by the organization in any way. Differently, the infection probability can be controlled with improvements at the level of technical defense mechanisms, employee training or changes in policy. Furthermore, additional insights can be drawn by performing the same sensitivity analysis procedure but fixating some of the variable inputs — particularly the ones considered uncontrollable — to average values observed historically in that specific organizational context.

To summarize, the modelling is intended to aid decision-makers in exploring the consequences of potential decisions. Here we have explored a model with some specific scenarios and, for decision-making, a company would need to parametrize the model with its situation in terms of office set-up, staff skill-levels, and helpdesk support levels and policies. However, even in the scenarios we have run, there are some general insights:

- Embedded recovery is valuable both for travellers with poor connectivity and in reducing recovery network loads within the office.
- Helpdesk and support policies can be critical in the smooth running of mass recovery, but need tailoring according to an organization's set-up and staff skill-levels.
- The model shows sensitivities to factors outside of the direct recovery solution, such as the probability of infection. Thus, when making recovery decisions, it is important to examine the wider enterprise systems such as AV, phishing protections, etc..

## Conclusion

With an estimated global cost of $20 billion in 2021 alone (19) that might or might not have been reduced during the Covid lockdown period, and an ever increasing list of high profile victims in both private and public sector organizations, ransomware is and will continue to be a problem in the future. That is why most organizations will have to understand and manage the risk of ransomware infection, or else face severe operational problems that could even turn to existential threats.

However, the tools available to support decision makers in better understanding the ransomware phenomena at the level of their own organization and guide the recovery technology selection, allocation and policy adaptation procedures, are not that many. Furthermore, most of the times, such tools exist in the form of general guidelines that do not offer enough practical understanding

to decision makers which are less security inclined. Given this, we have considered organizational recovery under ransomware as suitable for explicit, simulation modelling, due to descriptive power, modularity and possibility for extension, and ability to produce quantitative measures for further analysis.

In this paper, we have described our structured modelling conceptualization based on the so called Distributed Systems Metaphor and explained how it supports the model co-design methodology. We have exemplified the suitability of our approach in a complex, yet technology focused organizational environment with the implementation of the recovery model. We then used the model to generate several recovery scenarios corresponding to the behaviour manifested by ransomware strains and tested the impact of recovery techniques on organizational recovery time under such conditions. Furthermore, we have detailed the main verification and validation procedures employed, and how the model could be practically used in a decision-making context.

However, as with any model, the decisions regarding what not to model explicitly are just as relevant. Organisational resilience is a complex phenomenon: device recovery, the nature of attacks, technical security mechanisms, policy, budgets and user training and awareness and possibly even other 'unknowns' play important parts in overall resilience. Our models focus primarily on the performance and behaviour of device recovery mechanisms under different ransomware attacks. Although we could argue that elements of technical security mechanisms, policy, user training and awareness are present in the models at a very low level of detail — in the infection probability and need of admins parameters — a separate analysis and model would be required to reason about security in this context. For example, this could include an analysis of how images are protected; is there a mechanism for ensuring only approved images are installed and that their integrity can be validated (Section 2.2.4) and where images are maintained on the device, can they be protected from malware. Such security analysis and decisions should go alongside the organizational model when considering a robust recovery strategy.

Nevertheless, the current model configuration is well-suited for further development. On the practical side, model extensions could include explicit conceptualization and implementation of employee work — for example in the form of processes generating revenue —, data loss, technical security mechanisms and policies around data storage or increasing the level of detail for users and devices interacting with one another. From a theoretical perspective, the modelling formalism could be further extended at the level of supporting tools: for example, to support local reasoning about the compositional model structure or to provide dynamic model-checking capabilities. Last but not least, additional research can be done in the area of empirical case studies: to include models constructed using this approach and deployed in varied organizational environments, in an attempt for further methodological refinement.

## References


1. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439 (2017)
2. Abrams, L.: Jvckenwood hit by conti ransomware claiming theft of 1.5tb data. Bleepingcomputer (2021), https://www.bleepingcomputer.com/news/security/jvckenwood-hit-by-conti-ransomware-claiming-theft-of-15tb-data/. Accessed 05/07/2023
3. Abrams, L.: Allied universal breached by maze ransomware, stolen data leaked (2019), https://www.bleepingcomputer.com/news/security/allied-universal-breached-by-maze-ransomware-stolen-data-leaked/. Accessed 05/07/2023
4. Alwashali, A.A.M.A., Abd Rahman, N.A., Ismail, N.: A survey of ransomware as a service (raas) and methods to mitigate the attack. In: 2021 14th International Conference on Developments in eSystems Engineering (DeSE). pp. 92–96. IEEE (2021)
5. Analytica, O.: Us pipeline hack to make ransomware risks a priority. Emerald Expert Briefings (2021)
6. Anderson, G., D.Pym: A calculus and logic of bunched resources and processes. Theoretical Computer Science **614**, 63–96 (2016)
7. ATT&CK, M.: Mitre att&ck matrix for enterprise, https://attack.mitre.org/matrices/enterprise/. Accessed 05/07/2023
8. Baker, W., Nohria, N., Eccles, R.: The network organization in theory and practice. Classics of Organization Theory **8**, 401 (1992)
9. Baldwin, A., Beres, Y., Duggan, G.B., Mont, M.C., Johnson, H., Middup, C., Shiu, S.: Economic methods and decision making by security professionals. In: Schneier B. (eds) Economics of Information Security and Privacy III. Springer, New York, NY (2012), 978-1-4614-1981-5
10. Baldwin, A., Caulfield, T., Ilau, M.C., Pym, D.: Modelling organizational recovery. In: Simulation Tools and Techniques: 13th EAI International Conference, SIMUtools 2021, Virtual Event, November 5-6, 2021, Proceedings. pp. 284–314. Springer (2022)
11. Baroni, G., Francke, T.: An effective strategy for combining variance- and distribution-based global sensitivity analysis. Environmental Modelling & Software **134**, 104851 (2020). https://doi.org/10.1016/j.envsoft.2020.104851
12. Beres, Y., Griffin, J., Shiu, S., Heitman, M., Markle, D., Ventura, P.: Analysing the performance of security solutions to reduce vulnerability exposure window. 2008 Annual Computer Security Applications Conference (ACSAC) pp. 33–42 (2008)
13. Beresnevichiene, Y., Pym, D., Shiu, S.: Decision support for systems security investment. 2010 IEEE/IFIP Network Operations and Management Symposium Workshops pp. 118–125 (2010)
14. Binary Defense: Emotet Evolves With new Wi-Fi Spreader. Available at https://www.binarydefense.com/emotet-evolves-with-new-wi-fi-spreader/. Accessed 11/07/2023 (2020)
15. Birtwistle, G.: Demos — discrete event modelling on Simula. Macmillan (1979)
16. Birtwistle, G.: Demos implementation guide and reference manual. Tech. Rep. 81/70/22, University of Calgary (1981)
17. [blinded]: SysModels Julia Package. Available at [blinded]. Accessed 10/05/2021
18. Boyanov, P.: Educational exploiting the information resources and invading the security mechanisms of the operating system windows 7 with the exploit eternalblue and backdoor doublepulsar. Association Scientific and Applied Research **14**, 34 (2018)
19. Braue, D.: Global ransomware damage costs predicted to exceed $265 billion by 2031 (2022), https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-250-billion-usd-by-2031/. Accessed 11/07/2023
20. Brierley, C., Pont, J., Arief, B., Barnes, D.J., Hernandez-Castro, J.: Persistence in linux-based iot malware. In: Nordic Conference on Secure IT Systems. pp. 3–19. Springer (2020)
21. Broadhead, S.: The contemporary cybercrime ecosystem: A multi-disciplinary overview of the state of affairs and developments. Computer Law & Security Review **34**(6), 1180–1196 (2018)
22. Bromiley, M.: SANS 2019 Incident Response(IR) Survey: It's Time for aChange. SANS (2012), https://www.sans.org/reading-room/whitepapers/analyst/2019-incident-response-ir-survey-time-change-39070. Accessed 28/06/2021
23. Bronk, C., Tikk-Ringas, E.: The cyber attack on saudi aramco. Survival **55**(2), 81–96 (2013)
24. Burns, T., Stalker, G.M.: Mechanistic and organic systems. Classics of organizational theory pp. 209–214 (1961)
25. Caulfield, T., Demjaha, A., Pym, D.: Found in translation: Co-design for security modelling. In: Proc. 11th STAST 2021. Springer (2021)
26. Caulfield, T., Pym, D.: Improving security policy decisions with models. IEEE Security and Privacy **13(5)**, 34–41 (2015)
27. Caulfield, T., Pym, D.: Modelling and simulating systems security policy. In: Proc. SimuTools (2015)
28. Caulfield, T., Ilau, M.C., Pym, D.: Meta-modelling for ecosystems security. In: Proc 13th SIMUtools 2021. Springer (2021)
29. Caulfield, T., Ilau, M.C., Pym, D.: Engineering ecosystem models: Semantics and pragmatics. In: International Conference on Simulation Tools and Techniques. pp. 236–258. Springer (2022)
30. Caulfield, T., Ilau, M.C., Pym, D.: Meta-modelling for ecosystems security. In: International Conference on Simulation Tools and Techniques. pp. 259–283. Springer (2022)
31. Chad, H.: Malware lateral movement: A primer. Mandiant (2015), https://www.mandiant.com/resources/malware-lateral-move. Accessed 12/07/2023
32. Chakkaravarthy, S.S., Sangeetha, D., Vaidehi, V.: A survey on malware analysis and mitigation techniques. Computer Science Review **32**, 1–23 (2019)
33. Charlie, O.: Updated kaseya ransomware attack faq: What we know now (2021), https://www.zdnet.com/article/updated-kaseya-ransomware-attack-faq-what-we-know-now/. Accessed 12/07/2023
34. Chervenak, A., Vellanki, V., Kurmas, Z.: Protecting file systems: A survey of backup techniques. In: Joint NASA and IEEE Mass Storage Conference. vol. 99. Citeseer (1998)
35. cisa : Emotet Malware. Available at https://us-cert.cisa.gov/ncas/alerts/aa20-280a. Accessed 28/06/2021 (2017)

36. CISA: Destructive malware targeting organizations in ukraine. National Cyber Awareness System (2022), https://www.cisa.gov/uscert/ncas/alerts/aa22-057a. Accessed 05/07/2023
37. Collinson, M., Monahan, B., Pym, D.: A discipline of mathematical systems modelling (2011), to appear: College Publications, London
38. Collinson, M., Monahan, B., Pym, D.: A Discipline of Math.Systems Modelling. College Publns. (2012)
39. Collinson, M., Pym, D.: Algebra and logic for resource-based systems modelling. Math. Structures in Comput. Sci. **19**, 959–1027 (2009)
40. Collinson, M., Monahan, B., Pym, D.: Semantics for structured systems modelling and simulation. In: Proc. Simutools 2010. ACM Digital Library, ISBN 78-963-9799-87-5 (2010)
41. Crainer, S.: Key management ideas: Thinkers that changed the management. World Pretince Hall Books. New York (1993)
42. Dahl, O.J., Nygaard, K.: Simula: an algol-based simulation language. Communications of the ACM **9**(9), 671–678 (1966)
43. Dansimp, mjcaparas, v jmathew, MSFTTracyP, JoeDavies-MSFT, alexbuckgit: What is ransomware. Microsoft Security (2022), https://docs.microsoft.com/en-us/security/compass/human-operated-ransomware. Accessed 12/07/2023
44. David, S., Sabiescu, A.G., Cantoni, L.: Co-design with communities. a reflection on the literature. In: Proceedings of the 7th International Development Informatics Association Conference. pp. 152–166. No. 2013, IDIA Pretoria, South Africa (2013)
45. Dehlawi, Z., Abokhodair, N.: Saudi arabia's response to cyber conflict: A case study of the shamoon malware incident. In: 2013 IEEE International Conference on Intelligence and Security Informatics. pp. 73–75. IEEE (2013)
46. Demos2k: sourceforge.net, the Demos2k distribution is part of the Seymour distribution
47. Eccles, R.G., Crane, D.B.: Managing through networks in investment banking. California management review **30**(1), 176–195 (1987)
48. Eckhardt, R.: Stan ulam, john von neumann, and the monte carlo method. Los Alamos Science **15**(131-136), 30 (1987)
49. Edward, K.: What is ransomware as a service (raas)? the dangerous threat to world security (2022), https://www.upguard.com/blog/what-is-ransomware-as-a-service. Accessed 12/07/2023.
50. Eric Parizo: Maersk CISO Says NotPeyta Devastated Several Unnamed US firms. Available at https://www.darkreading.com/threat-intelligence/maersk-ciso-says-notpeyta-devastated-several-unnamed-us-firms/a/d-id/1336558?page_number=2. Accessed 28/06/2021 (2019)
51. Europol: Internet organised crime threat assessment (2020), https://www.europol.europa.eu/cms/sites/default/files/documents/internet_organised_crime_threat_assessment_iocta_2020.pdf. Accessed 12/07/2023
52. F-Secure: Trojan:androidkoler. F-Secure (2013), https://www.f-secure.com/v-descs/trojan_android_koler.shtml. Accessed 05/07/2023
53. FBI: 2019 internet crime report (2020), https://www.fbi.gov/news/stories/2019-internet-crime-report-released-021120. Accessed 11/07/2023
54. FBI: Cisa-fbi guidance for msps and their customers affected by the kaseya vsa supply-chain ransomware (2021), https://www.cisa.gov/uscert/ncas/current-activity/2021/07/04/cisa-fbi-guidance-msps-and-their-customers-affected-kaseya-vsa. Accessed 11/07/2023
55. Forrester, J.W.: Industrial dynamics. Journal of the Operational Research Society **48**(10), 1037–1041 (1997)
56. Galison, P., et al.: Image and logic: A material culture of microphysics. University of Chicago Press (1997)
57. Galmiche, D., Méry, D., Pym, D.: The Semantics of BI and Resource Tableaux. Math. Structures in Comput. Sci. **15**, 1033–1088 (2005)
58. Gareth, M.: Images de l'organisation. Presses de l'Université Laval (2019)
59. Gibson, C.A., Tarrant, M.: A'conceptual models' approach to organisational resilience. Australian Journal of Emergency Management, The **25**(2), 6–12 (2010)
60. Gilmore, S., Hillston, J.: The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In: Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation. pp. 352–368. No. 794 in Lecture Notes in Computer Science, Springer-Verlag (1994)
61. Gittins, Z., Soltys, M.: Malware persistence mechanisms. Procedia Computer Science **176**, 88–97 (2020)
62. Government, U.: Cyber security breaches survey 2020 (2020), https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020/cyber-security-breaches-survey-2020. Accessed 17/07/2023
63. GROUP, I.: Overview of the 9 distinct data wipers used in the ukraine war. RecordedFuture (2022), https://www.recordedfuture.com/overview-9-district-data-wipers-ukraine-war. Accessed 17/07/2023
64. Gulati, R., Nohria, N., Zaheer, A.: Strategic networks. Strategic management journal **21**(3), 203–215 (2000)
65. Hearn, A.: Hackers publish private photos from cosmetic surgery clinic. The Guardian (2017), https://www.theguardian.com/technology/2017/may/31/hackers-publish-private-photos-cosmetic-surgery-clinic-bitcoin-ransom-payments. Accessed 17/07/2023
66. Hewlett-Packard Laboratories: Security Analytics. Available at https://www.hpl.hp.com/news/2011/oct-dec/security_analytics.html. Accessed 10/06/2023
67. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall International, London (1985)
68. Hobbs, A.: The colonial pipeline hack: Exposing vulnerabilities in us cybersecurity. In: SAGE Business Cases. SAGE Publications: SAGE Business Cases Originals (2021)
69. Hole, K.J.: Robustness to malware reinfections. In: Anti-fragile ICT Systems, pp. 93–98. Springer (2016)
70. Hopkins, M., Dehghantanha, A.: Exploit kits: The production line of the cybercrime economy? In: 2015 second international conference on Information Security and Cyber Forensics (InfoSec). pp. 23–27. IEEE (2015)
71. Houghton, F.: Cybersecurity, ransomware attacks and health: Exploring the public health implications of the recent cyberattack on ireland's health service. Medicina Internacia Revuo **29**(116), 160–163 (2021)

72. HP: HP SureRecover. HP (2021), https://www8.hp.com/h20195/v2/GetPDF.aspx/4AA7-4556ENW.pdf. Accessed 16/07/2023
73. Iliev, A., Kyurkchiev, N., Rahnev, A., Terzieva, T.: Some new approaches for modelling large-scale worm spreading on the internet. ii. Neural, Parallel, and Scientific Computations **27**(1), 23–34 (2019)
74. Ioannidis, C., Pym, D., Williams, J., Gheyas, I.: Resilience in information stewardship. European journal of operational research **274**(2), 638–653 (2019)
75. Ishtiaq, S., O'Hearn, P.: BI as an assertion language for mutable data structures. In: Proc. POPL (2001)
76. Karapapas, C., Pittaras, I., Fotiou, N., Polyzos, G.C.: Ransomware as a service using smart contracts and ipfs. arXiv preprint arXiv:2003.04426 (2020)
77. Karyotis, V., Khouzani, M.: Chapter 3 - early malware diffusion modeling methodologies. In: Karyotis, V., Khouzani, M. (eds.) Malware Diffusion Models for Wireless Complex Networks, pp. 39–60. Morgan Kaufmann, Boston (2016). https://doi.org/https://doi.org/10.1016/B978-0-12-802714-1.00013-X
78. Keijzer, N.: The new generation of ransomware: an in depth study of Ransomware-as-a-Service. Master's thesis, University of Twente (2020)
79. Kerns, Q., Payne, B., Abegaz, T.: Double-extortion ransomware: A technical analysis of maze ransomware. In: Proceedings of the Future Technologies Conference. pp. 82–94. Springer (2021)
80. Killijian, M.O., Courtès, L., Powell, D.: A survey of cooperative backup mechanisms (2006)
81. Kolmogorov, A.: Sulla determinazione empirica di una lgge di distribuzione. Inst. Ital. Attuari, Giorn. **4**, 83–91 (1933)
82. Korzybski, A.: Science and sanity: An introduction to non-Aristotelian systems and general semantics. Institute of GS (1958)
83. Kraemer-Mbula, E., Tang, P., Rush, H.: The cybercrime ecosystem: Online innovation in the shadows? Technological Forecasting and Social Change **80**(3), 541–555 (2013)
84. Kral, P.: The Incident Handlers Handbook. SANS (2012), https://www.sans.org/reading-room/whitepapers/incident/incident-handlers-handbook-33901. Accessed 16/07/2023
85. Krebs, B.: Inside a reveton ransomware operation. KrebsonSecurity, August (2012), https://krebsonsecurity.com/2012/08/inside-a-reveton-ransomware-operation/. Accessed 05/07/2023
86. Kshetri, N., Voas, J.: Do crypto-currencies fuel ransomware? IT professional **19**(5), 11–15 (2017)
87. Lewinson, E.: Violin plots explained (2019), https://towardsdatascience.com/violin-plots-explained-fb1d115e023d. Accessed 16/07/2023
88. McColl, J.: Probability. Elsevier: Butterworth–Heinemann (1995)
89. Meland, P.H., Bayoumy, Y.F.F., Sindre, G.: The ransomware-as-a-service economy within the darknet. Computers & Security **92**, 101762 (2020)
90. Microsoft: Windows Recovery Environment (Windows RE). Microsoft (2017), https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/windows-recovery-environment--windows-re--technical-reference. Accessed 28/06/2021
91. Microsoft: Apply features and settings on your devices using device profiles in Microsoft Intune. Microsoft (2020), https://docs.microsoft.com/en-us/mem/intune/configuration/device-profiles. Accessed 28/06/2021
92. Microsoft: Deploy Windows 10 using PXE and Configuration Manager. Microsoft (2021), https://docs.microsoft.com/en-us/windows/deployment/deploy-windows-cm/deploy-windows-10-using-pxe-and-configuration-manager. Accessed 28/06/2021
93. Microsoft: Microsoft Deployment Toolkit. Microsoft (2021), https://docs.microsoft.com/en-us/windows/deployment/deploy-windows-mdt/get-started-with-the-microsoft-deployment-toolkit. Accessed 28/06/2021
94. Microsoft: Overview of Windows Autopilot. Microsoft (2021), https://docs.microsoft.com/en-us/mem/autopilot/windows-autopilot. Accessed 28/06/2021
95. Microsoft: Destructive malware targeting ukrainian organizations. Microsoft Security (2022), https://www.microsoft.com/security/blog/2022/01/15/destructive-malware-targeting-ukrainian-organizations/. Accessed 16/07/2023
96. Milletary, J.: Citadel trojan malware analysis. Dell SecureWorks (2012)
97. Milner, R.: A Calculus of Communicating Systems, LNCS, vol. 92. Springer Verlag (1980)
98. Milner, R.: Calculi for synchrony and asynchrony. Theor. Comput. Sci. **25**(3), 267–310 (1983)
99. Milner, R.: Communicating and mobile systems: the $\pi$-calculus. Cambridge University Press (1999)
100. Milner, R.: Bigraphs as a model for mobile interaction (invited paper). In: ICGT 2002, First International Conference on Graph Transformation. LNCS, vol. 2505, pp. 8–13. Springer (2002)
101. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press (2009). https://doi.org/10.1017/CBO9780511626661
102. Mimoso, M.: Europol takes down ransomware gang in spain, uae. Threatpost, February (2013), https://threatpost.com/europol-takes-down-ransomware-gang-spain-uae-021413/77529/. Accessed 16/07/2023
103. Mintzberg, H.: The structuring of organizations. Englewood Cliffs **330** (1979)
104. Mishra, B.K., Jha, N.: Seiqrs model for the transmission of malicious objects in computer network. Applied Mathematical Modelling **34**(3), 710–715 (2010)
105. Moran Stritch, M., Winterburn, M., Houghton, F.: The conti ransomware attack on healthcare in ireland: Exploring the impacts of a cybersecurity breach from a nursing perspective. Canadian Journal of Nursing Informatics **16**(3-4) (2021)
106. Morgan, J.M., Liker, J.K.: The Toyota product development system: integrating people, process, and technology. Productivity press (2020)
107. Nuce, J., Kennelly, J., Goody, K., Moore, A., Rahman, A., Williams, M., McKeague, B., Wilson, J.: Shining a light on darkside ransomware operations. FireEye Blogs (2021)
108. O'Hearn, P.: Resources, concurrency, and local reasoning. Theor. Comput. Sci. **375**(1–3), 271–307 (May 2007)

109. O'Hearn, P.W., Reynolds, J.C., Yang, H.: Local reasoning about programs that alter data structures. In: Proceedings of the 15th International Workshop on Computer Science Logic. p. 1–19. CSL '01, Springer-Verlag, Berlin, Heidelberg (2001)
110. O'Hearn, P., Pym, D.: The logic of bunched implications. Bulletin of Symbolic Logic **5(2)**, 215–244 (1999)
111. O'Kane, P., Sezer, S., Carlin, D.: Evolution of ransomware. Iet Networks **7**(5), 321–327 (2018)
112. Oz, H., Aris, A., Levi, A., Uluagac, A.S.: A survey on ransomware: Evolution, taxonomy, and defense solutions. ACM Computing Surveys (CSUR) (2021)
113. Pape, R.A.: Bombing to win. Cornell University Press (2014)
114. Pianosi, F., Wagener, T.: A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. Environmental Modelling & Software **67**, 1–11 (2015)
115. Pianosi, F., Wagener, T.: Distribution-based sensitivity analysis from a generic input-output sample. Environmental Modelling & Software **108**, 197–207 (2018)
116. Plotkin, G.D.: A structural approach to operational semantics. Tech. Rep. DAIMI FN-19, Computer Science Dept., Aarhus University, Aarhus, Denmark (1981)
117. Pym, D.: Resource semantics: Logic as a modelling technology. ACM SIGLOG News **6**(2), 5–41 (Apr 2019). https://doi.org/10.1145/3326938.3326940, https://doi.org/10.1145/3326938.3326940
118. Pym, D.: Resource semantics: logic as a modelling technology. ACM SIGLOG News **6**(2), 5–41 (2019)
119. Pym, D., O'Hearn, P., Yang, H.: Possible Worlds and Resources: The Semantics of BI. Theor. Comput. Sci. **315**(1), 257–305 (2004)
120. Raphael, S.: Up to 1,500 businesses affected by ransomware attack, u.s. firm's ceo says (2021), https://www.reuters.com/technology/hackers-demand-70-million-liberate-data-held-by-companies-hit-mass-cyberattack-2021-07-05/. Accessed 16/07/2023
121. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science. pp. 55–74. LICS '02, IEEE Computer Society, Washington, DC, USA (2002), http://dl.acm.org/citation.cfm?id=645683.664578
122. Rhodes, C., Bettany, A.: Windows Installation and Update Troubleshooting. Apress (2016)
123. Richardson, R., North, M.M.: Ransomware: Evolution, mitigation and prevention. International Management Review **13**(1), 10 (2017)
124. Schelling, T.C.: Arms and influence. Yale University Press (2008)
125. Shaun Hurley and Karan Sood: NotPetya Technical Analysis Part II: Further Findings and Potential for MBR Recovery. Available at https://www.crowdstrike.com/blog/petrwrap-technical-analysis-part-2-further-findings-and-potential-for-mbr-recovery/. Accessed 16/07/2023 (2017)
126. de Simone, R.: Higher-level synchronising devices in Meije-SCCS. Theor. Comput. Sci. **37**, 245–267 (1985)
127. Smirnov, N.V.: On the estimation of the discrepancy between empirical curves of distribution for two independent samples. Bull. Math. Univ. Moscou **2**(2), 3–14 (1939)
128. Sułkowski, Ł., et al.: Types of metaphors of organisation. Journal of Intercultural Management **3**(2), 221–227 (2011)
129. Szappanos, G.: Inside the blackhole. SophosLabs (2012)
130. Tweneboah-Kodua, S., Atsu, F., Buchanan, W.: Impact of cyberattacks on stock performance: a comparative study. Information & Computer Security (2018)
131. Umar, R., Riadi, I., Kusuma, R.S.: Analysis of conti ransomware attack on computer network with live forensic method. IJID (International Journal on Informatics for Development) **10**(1), 53–61 (2021)
132. Voinov, A., Jenni, K., Gray, S., Kolagani, N., Glynn, P.D., Bommel, P., Prell, C., Zellner, M., Paolisso, M., Jordan, R., et al.: Tools and methods in participatory modeling: Selecting the right tool for the job. Environmental Modelling & Software **109**, 232–255 (2018)
133. Vynck, G.D., Lerman, R., Nakashima, E., Alcantara, C.: The anatomy of a ransomware attack. Washington Post (2021), https://www.washingtonpost.com/technology/2021/07/09/how-ransomware-attack-works/?itid=mr_innovations_1. Accessed 16/07/2023
134. Wardle, P.: Methods of malware persistence on mac os x. In: Proceedings of the virus bulletin conference (2014)
135. Weisberg, M.: Simulation and similarity: Using models to understand the world. Oxford University Press (2012)
136. Wyke, J.: What is zeus? Sophos, May (2011)
137. Xiao, X., Fu, P., Dou, C., Li, Q., Hu, G., Xia, S.: Design and analysis of seiqr worm propagation model in mobile internet. Communications in Nonlinear Science and Numerical Simulation **43**, 341–350 (2017). https://doi.org/10.1016/j.cnsns.2016.07.012
138. Xiong, W., Legrand, E., Åberg, O., Lagerström, R.: Cyber security threat modeling based on the mitre enterprise att&ck matrix. Software and Systems Modeling **21**(1), 157–177 (2022)
139. Yang, H., O'Hearn, P.: A semantic basis for local reasoning. In: Nielsen, M., Engberg, U. (eds.) Foundations of Software Science and Computation Structures. pp. 402–416. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
140. Young, A., Yung, M.: Cryptovirology: Extortion-based security threats and countermeasures. In: Proceedings of the 1996 IEEE Conference on Security and Privacy. p. 129–140. SP'96, IEEE Computer Society, USA (1996)
141. Yuryna Connolly, L., Wall, D.S., Lang, M., Oddson, B.: An empirical study of ransomware attacks on organizations: an assessment of severity and salient factors affecting vulnerability. Journal of Cybersecurity **6**(1) (12 2020). https://doi.org/10.1093/cybsec/tyaa023
142. Zio, E.: Monte carlo simulation: The method. In: The Monte Carlo simulation method for system reliability and risk analysis, pp. 19–58. Springer (2013)

## Appendix A: Ransomware Strains

This section contains information regarding the ransomware strains analysed for the purpose of constructing the conceptual ransomware model. Based on the overall goals and techniques employed in the attacks, the strains are clustered into five categories — locker, crypto locker, leakware, destructive ransomware and ransomware as a service —, with the mention that some ransomware can fit into multiple categories at the same time. Therefore, these categories should be understood rather as operational capabilities of the ransomware, than completely separable classes. Furthermore, we took into account the entry point and type of network spreading once infection has been achieved.

| Name | Year | Locker | Crypto Locker | Leakware | Destructive | RaaS | Entry Point | Automated Spread |
|---|---|---|---|---|---|---|---|---|
| **PC Cyborg** | 1989 | X | | | | | Floppy disks | None |
| **Alien (Young & Yung)** | 1996 | | X | | | | None | None |
| **Young Leakware** | 2003 | | | X | | | None | None |
| **Krotten** | 2004 | X | | | | | Website downloads | None |
| **PGPCoder** | 2005 | | X | | | | Website downloads<br>Phishing emails | None |
| **Archiveus** | 2006 | | X | | X | | Website downloads<br>Phishing emailss | None |
| **CryZip** | 2006 | | X | | | | Website downloads | None |
| **Reveton** | 2012 | X | X | | | | Website downloads<br>Phishing emails<br>Exploit Kits | None |
| **Shamoon StoneDrill** | 2012 | | | | X | | Phishing emails &<br>Human operation | Network shares |
| **CryptoLocker** | 2013 | | X | | | | Phishing emails<br>Botnet droppers | None |
| **CryptoWall TorrentLocker** | 2014 | | X | | | | Phishing emails<br>Website downloads<br>Exploit Kits | None |
| **Linux.Encoder KeRanger** | 2015 | | X | | | | Magento shopping<br>cart vulnerability | None |
| **RansomWeb** | 2015 | | X | | | | Manual exploit | None |
| **Fusob Small** | 2015 | X | X | | | | Website downloads<br>Exploit kits | None |
| **Erebus** | 2016 | | X | | | | Website downloads | None |
| **Petya** | 2016 | | X | | | | Phishing emails<br>MeDoc exploit | Credential harvesting<br>EternalBlue<br>Worm like |
| **SamSam** | 2016 | | X | | | | JexBoss Exploit Kit<br>RDP with stolen<br>or brute force credentials | None |
| **WannaCry** | 2017 | | X | | | | Phishing emails<br>Scanning TCP port 445<br>— EternalBlue<br>DoublePulsar backdoor | EternalBlue<br>DoublePulsar<br>Worm like |
| **NotPetya** | 2017 | | X | | X | | MeDoc exploit, EternalBlue<br>Eternal Romance, Phishing emails | Credential harvesting,<br>token impersonation<br>EternalBlue, EternalRomance<br>Worm like |
| **BadRabbit** | 2017 | | X | | | | Website downloads | Credential harvesting,<br>dictionary attacks,<br>network shares,<br>EternalSynergy, Worm Like |
| **Maze** | 2019 | | X | X | | | Phishing emails<br>Website downloads<br>Exploit kits<br>Citrix web gateway RDP | None (human operated,<br>various exploits are used,<br>but manually) |
| **RobbinHood** | 2019 | | X | | | | RDP (brute force)<br>Exploit kits | None (human operated,<br>various exploits are used,<br>but manually) |
| **Conti** | 2019 | | X | X | X | X | TrickBot Malware<br>Spear Phishing<br>Human Operation(<br>Buying access<br>RDP with stolen or<br>brute force credentials) | None (human operated,<br>various exploits are used,<br>but manually) |
| **Darkside** | 2020 | | X | X | | X | Phishing emails<br>Human operated (<br>RDP with stolen<br>or brute forced credentials<br>Attacks on Virtual Desktop<br>Infrastructure (VDI)) | None (human operated,<br>various exploits are used,<br>but manually) |
| **Netwalker** | 2020 | | X | X | | X | Phishing emails<br>Human operated (<br>RDP with stolen<br>or brute forced credentials<br>Attacks on Virtual Desktop<br>Infrastructure (VDI)<br>Pulse Secure VPN exploit<br>Telerik UI exploit) | None (human operated,<br>various exploits are used,<br>but manually) |

**Table 1.** Ransomware Strains

## Appendix B: Model Diagrams

This appendix contains the conceptual diagrams related to the architecture and behaviour of the constructed models: Network Model, Malware Model and Server Model. The device model is being presented in the main text.

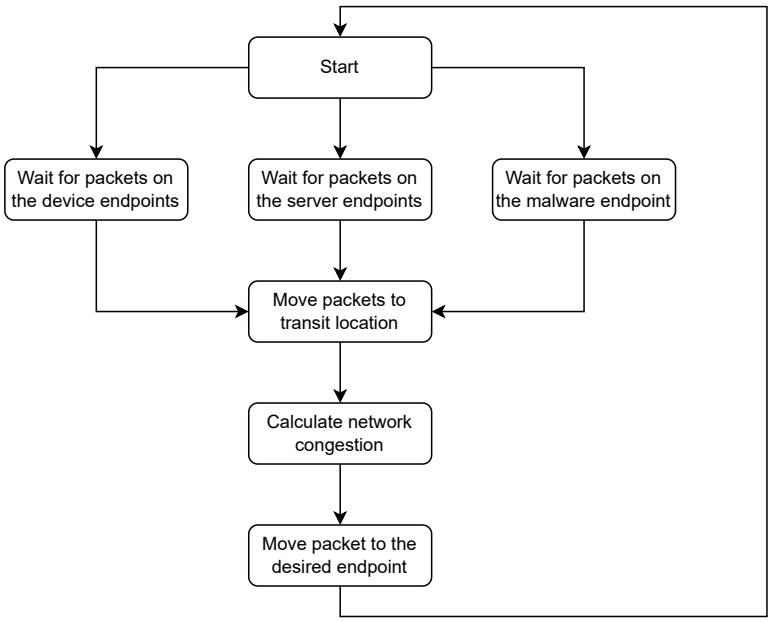



**Fig. 18.** Network Model Process Diagram

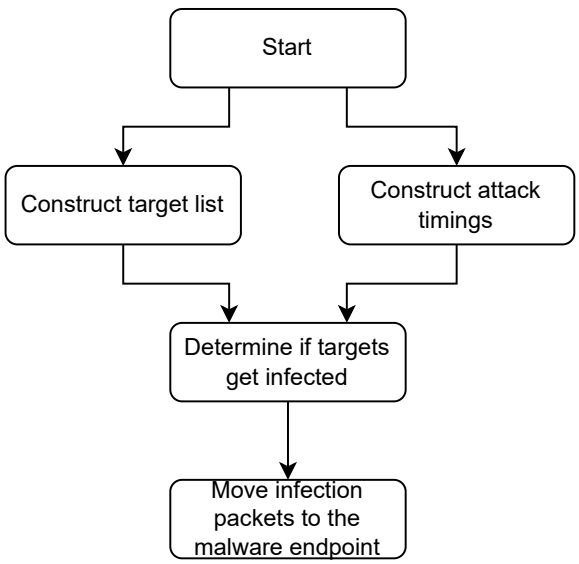



**Fig. 19.** Malware Model Process Diagram

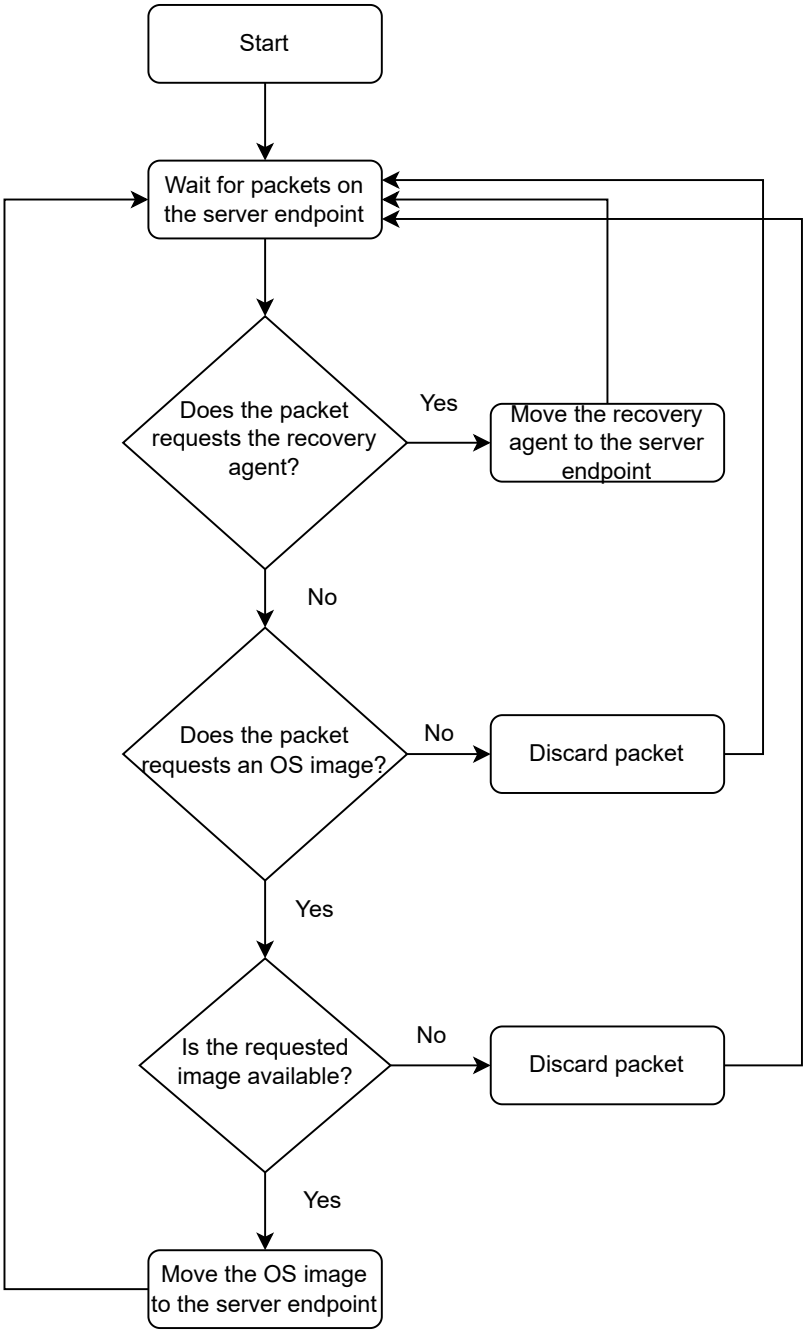



**Fig. 20.** Server Model Process Diagram

## Appendix C: Sensitivity Analysis & Results

This appendix contains information regarding the overall model execution and sensitivity analysis. Tables 2 to 4 contain the numerical outputs of the PAWN Sensitivity Analysis. Figures 21 to 23 contain the graphical representation of the sensitivity analysis. Figures 24 to 35 show additional experimentation performed: we analysed the possible impact of the variable input parameters — infection probability, attack duration, need of admin help and number of admins — on the output parameters — average recovery duration, total recovery duration and average number of devices recovering. We note here that additional experimentation could have been performed at the level of the recovery method: for example, by fixing one recovery method and another variable and then performing sensitivity analysis again.

| | Minimum | Mean | Median | Maximum | CV |
|---|---|---|---|---|---|
| Recovery Method | 0.280556 | 0.501074 | 0.553111 | 0.669556 | 0.325332 |
| Attack Type | 0.092444 | 0.134972 | 0.144444 | 0.158556 | 0.189384 |
| Infection Probability | 0.023667 | 0.080861 | 0.077111 | 0.145556 | 0.534977 |
| Attack Duration | 0.032667 | 0.063111 | 0.063111 | 0.093556 | 0.482394 |
| Admins Number | 0.018333 | 0.039289 | 0.027222 | 0.095000 | 0.715312 |
| Admins Need | 0.065444 | 0.106944 | 0.104222 | 0.153889 | 0.294688 |

**Table 2.** Sensitivity Analysis on the average recovery duration

| | Minimum | Mean | Median | Maximum | CV |
|---|---|---|---|---|---|
| Recovery Method | 0.024222 | 0.031259 | 0.027556 | 0.042000 | 0.246833 |
| Attack Type | 0.100556 | 0.142000 | 0.146444 | 0.174556 | 0.187439 |
| Infection Probability | 0.307556 | 0.531000 | 0.508278 | 0.799889 | 0.337688 |
| Attack Duration | 0.072333 | 0.096167 | 0.096167 | 0.120000 | 0.247834 |
| Admins Number | 0.005111 | 0.006422 | 0.006333 | 0.007889 | 0.165729 |
| Admins Need | 0.007111 | 0.009639 | 0.009333 | 0.012778 | 0.228576 |

**Table 3.** Sensitivity Analysis on the average number of devices recovering

| | Minimum | Mean | Median | Maximum | CV |
|---|---|---|---|---|---|
| Recovery Method | 0.151222 | 0.275815 | 0.279667 | 0.396556 | 0.363265 |
| Attack Type | 0.088333 | 0.133667 | 0.142000 | 0.162333 | 0.208527 |
| Infection Probability | 0.145556 | 0.305056 | 0.243000 | 0.588667 | 0.552403 |
| Attack Duration | 0.043778 | 0.073500 | 0.073500 | 0.103222 | 0.404384 |
| Admins Number | 0.010667 | 0.029622 | 0.020889 | 0.074000 | 0.764506 |
| Admins Need | 0.023778 | 0.051000 | 0.046778 | 0.086667 | 0.444535 |

**Table 4.** Sensitivity Analysis on the total recovery duration

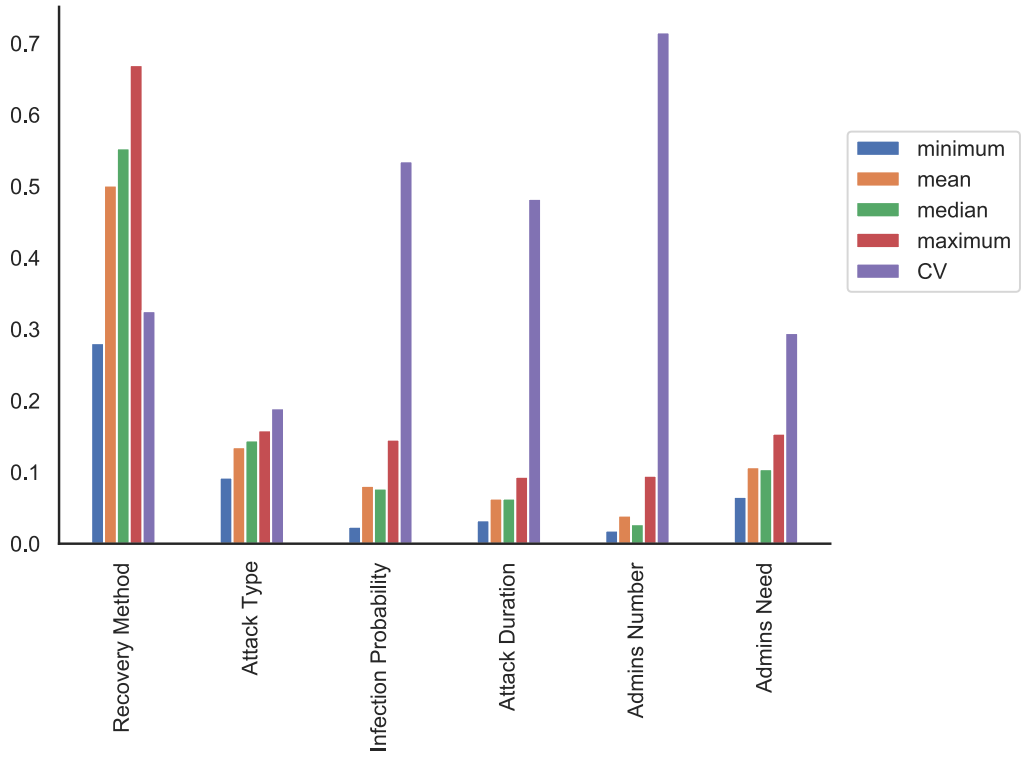



**Fig. 21.** Sensitivity Analysis on the average recovery duration of devices

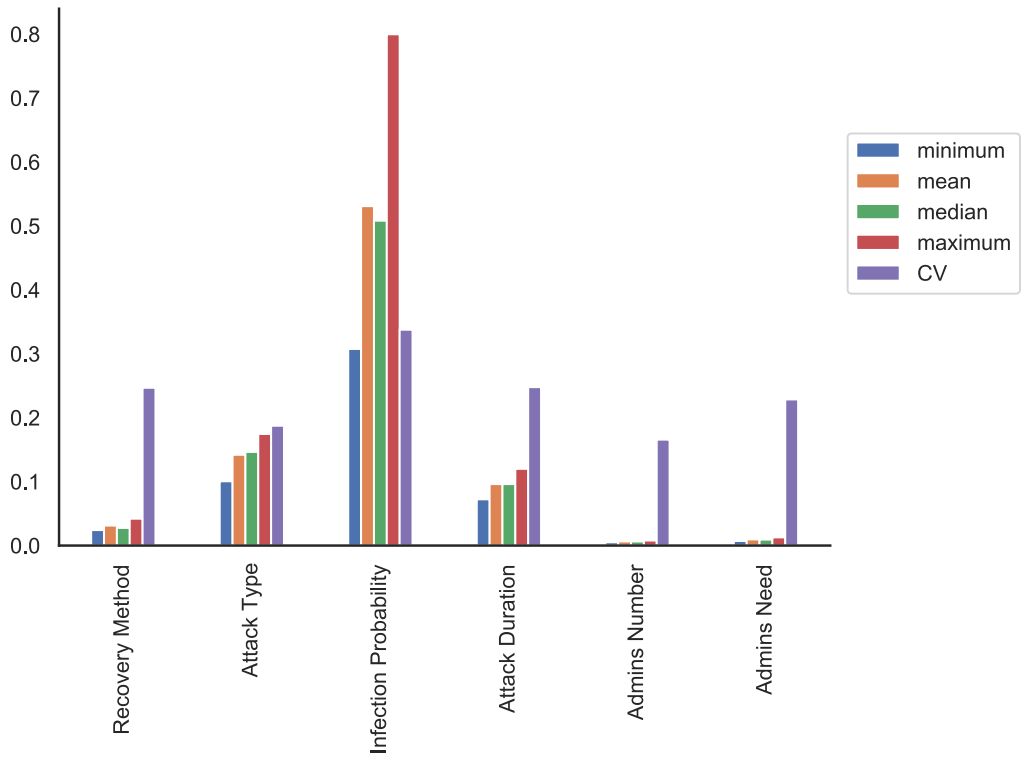



**Fig. 22.** Sensitivity Analysis on the average number of devices recovering

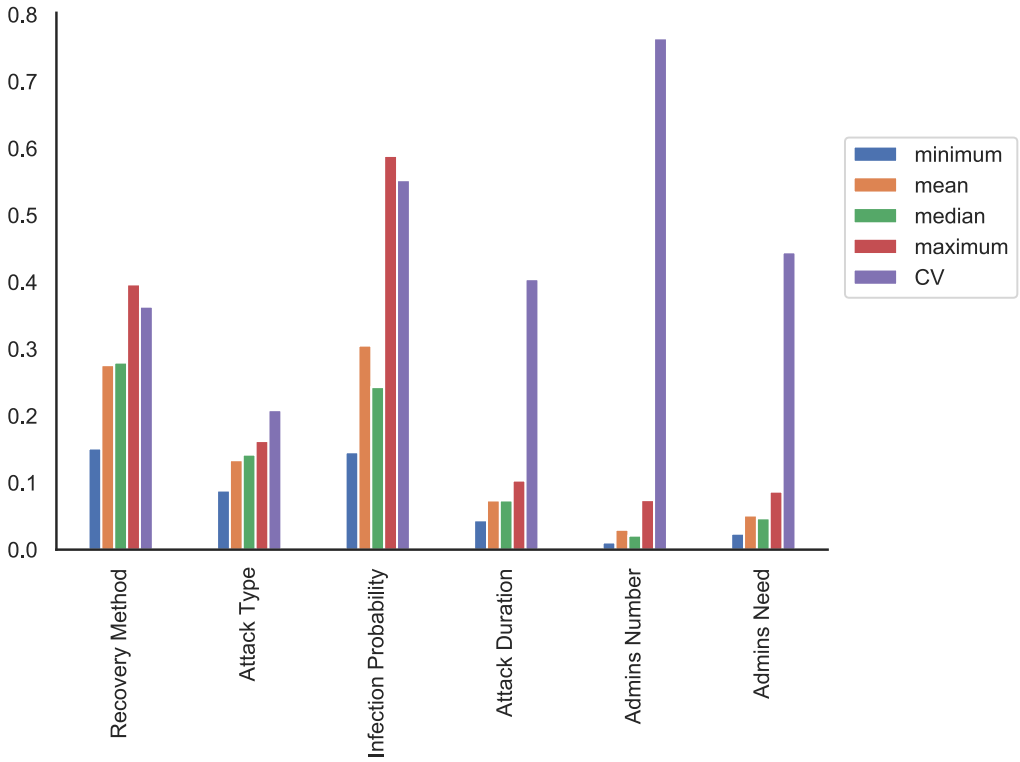



**Fig. 23.** Sensitivity Analysis on total recovery duration of devices

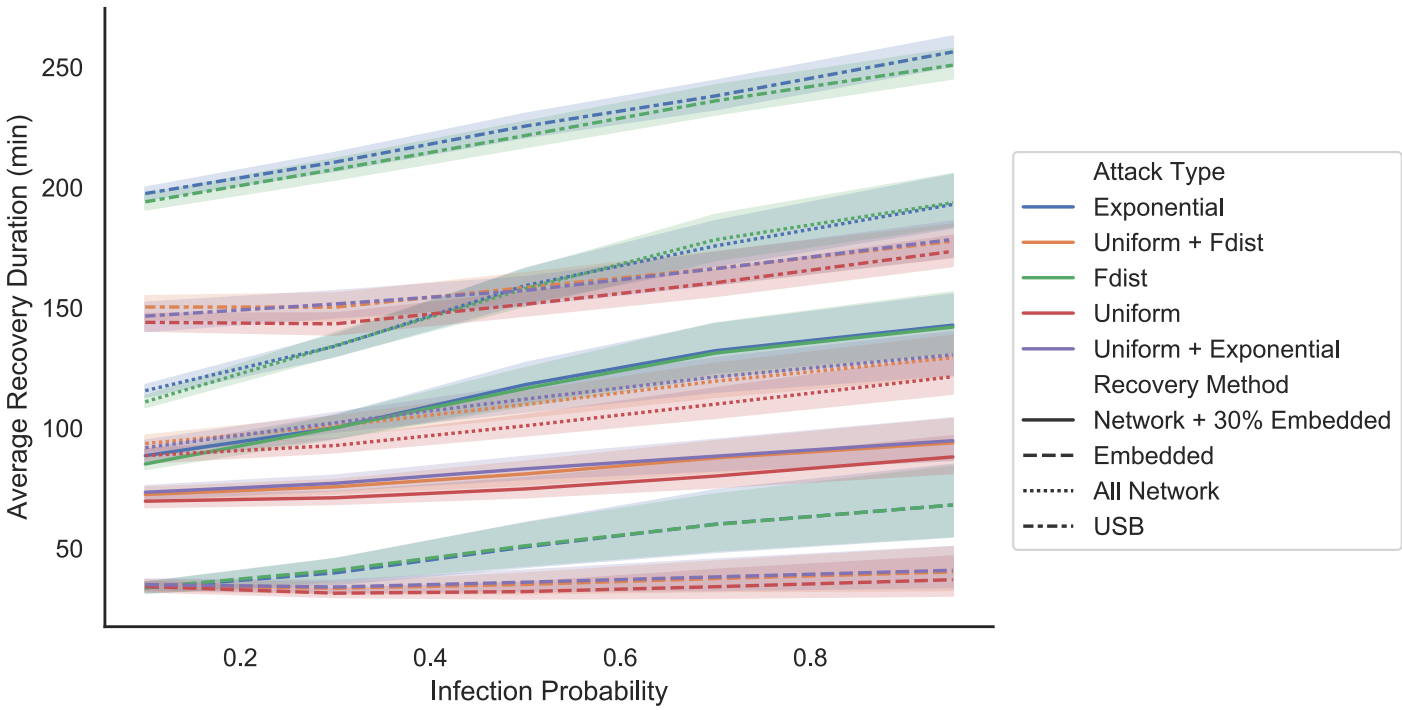



**Fig. 24.** Impact of infection probability on average recovery duration

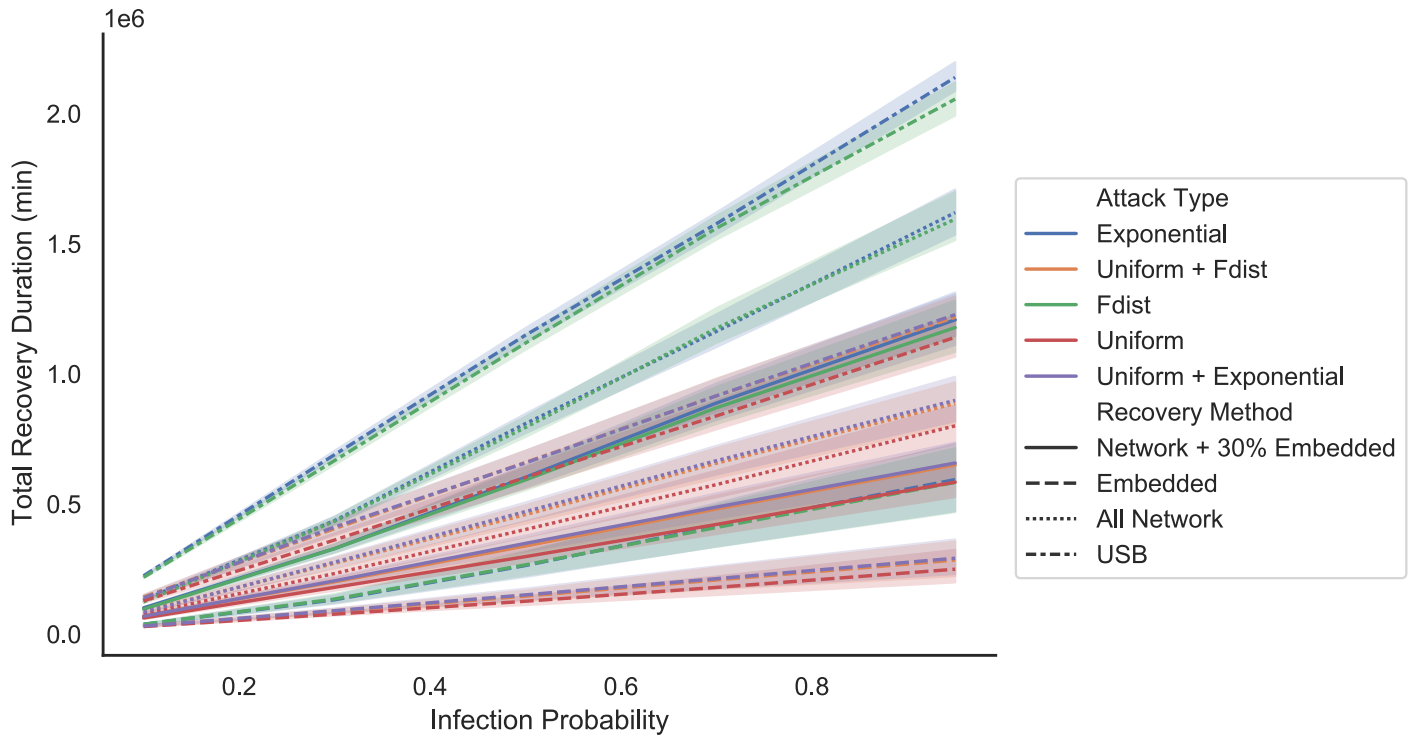



**Fig. 25.** Impact of infection probability on total recovery duration

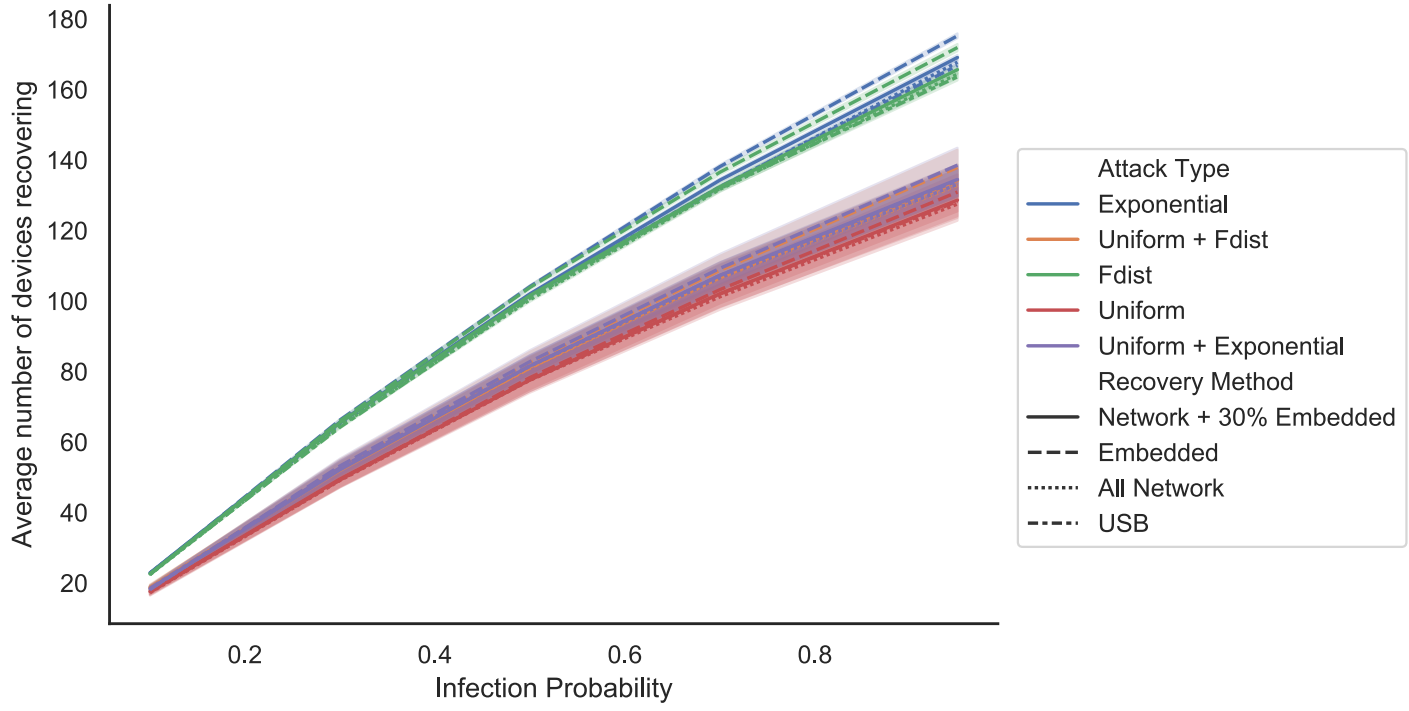



**Fig. 26.** Impact of infection probability on average number of devices recovering

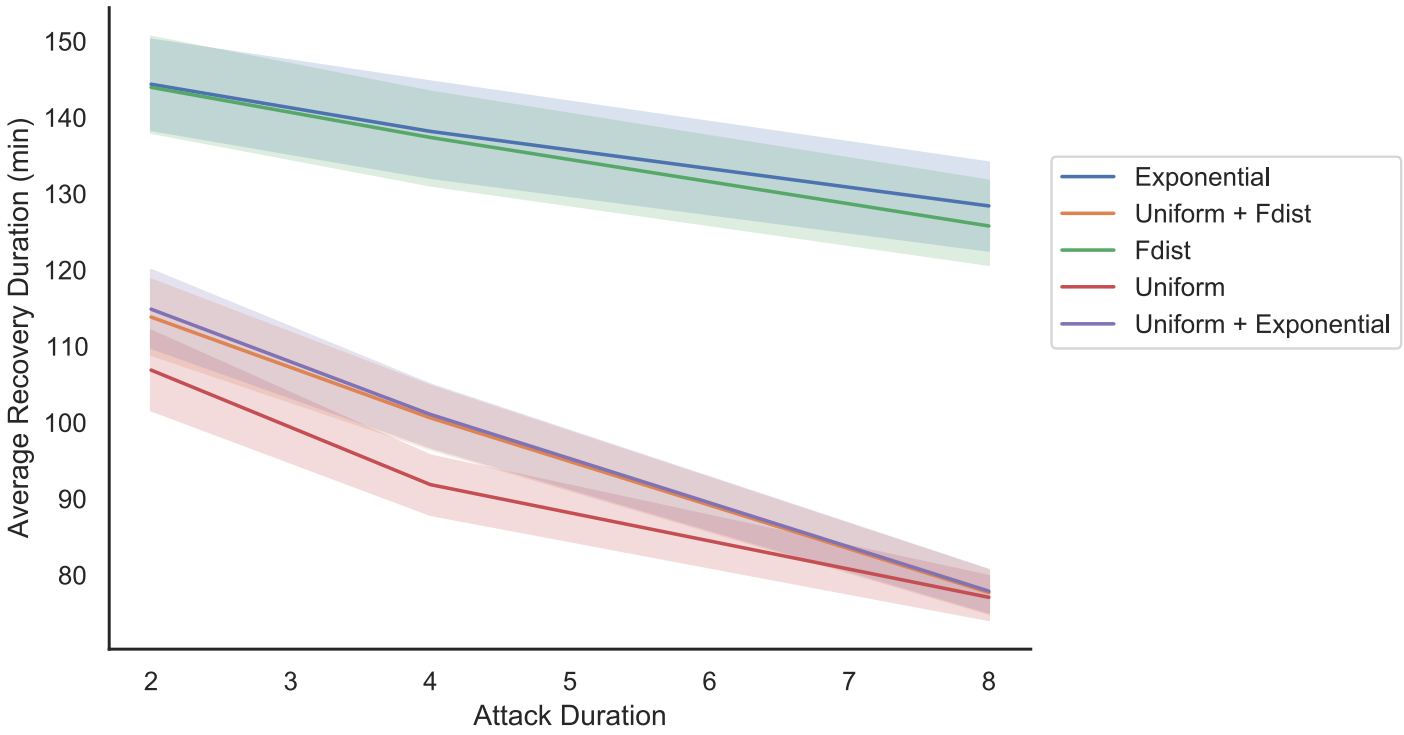



**Fig. 27.** Impact of attack duration on average recovery duration

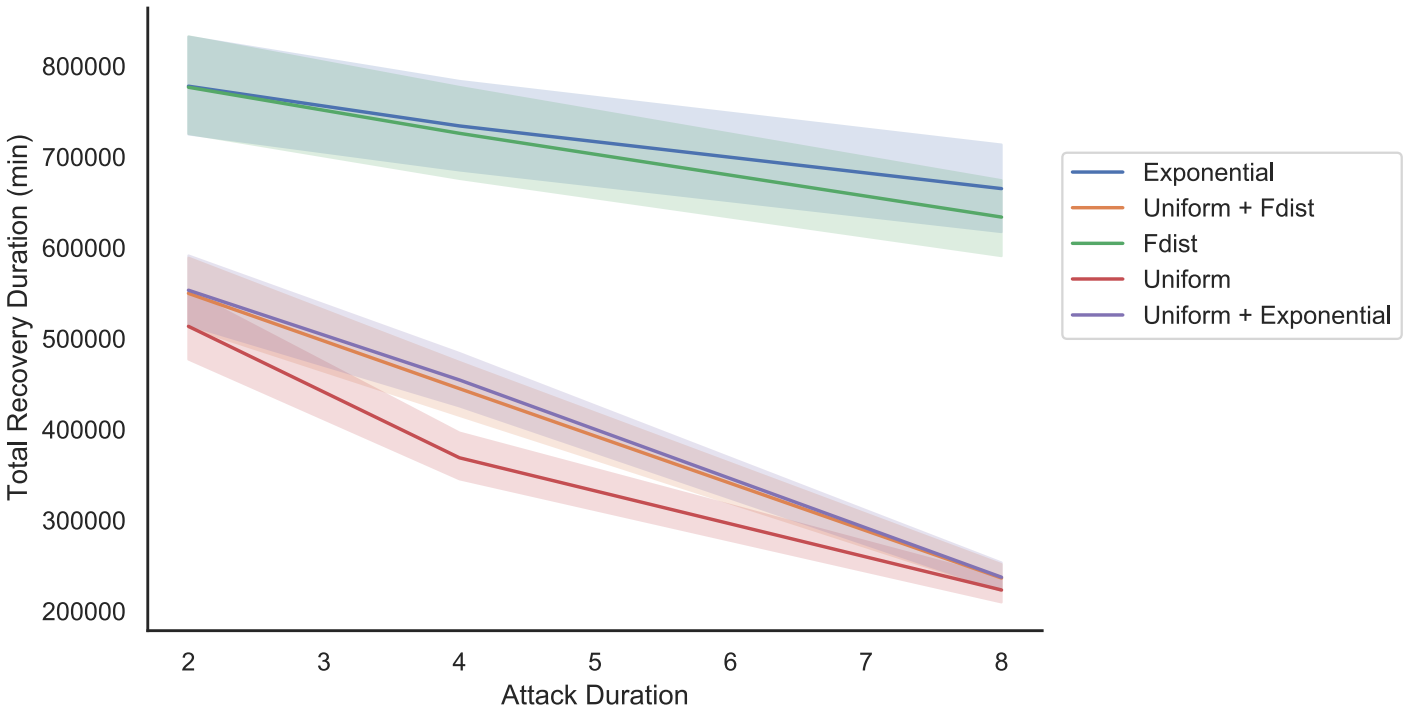



**Fig. 28.** Impact of attack duration on total recovery duration

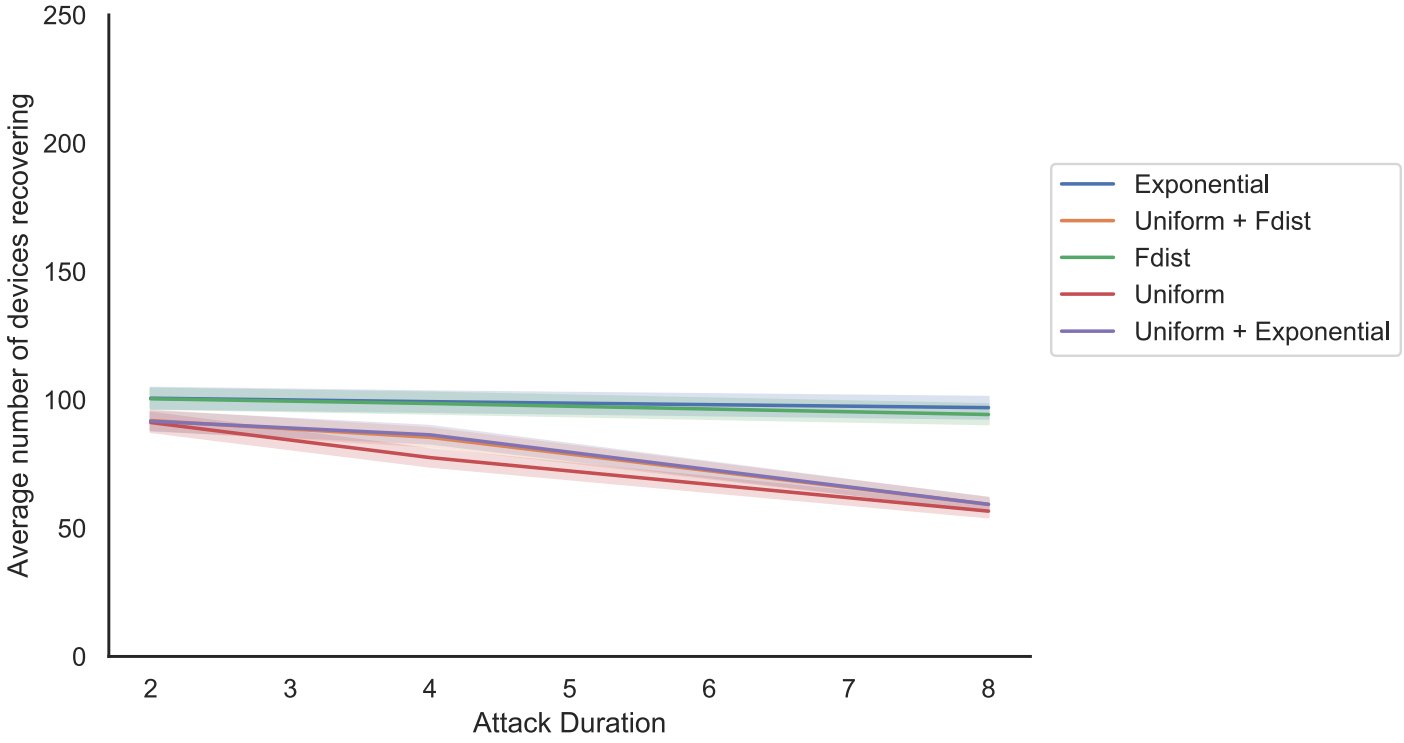



**Fig. 29.** Impact of attack duration on average number of devices recovering

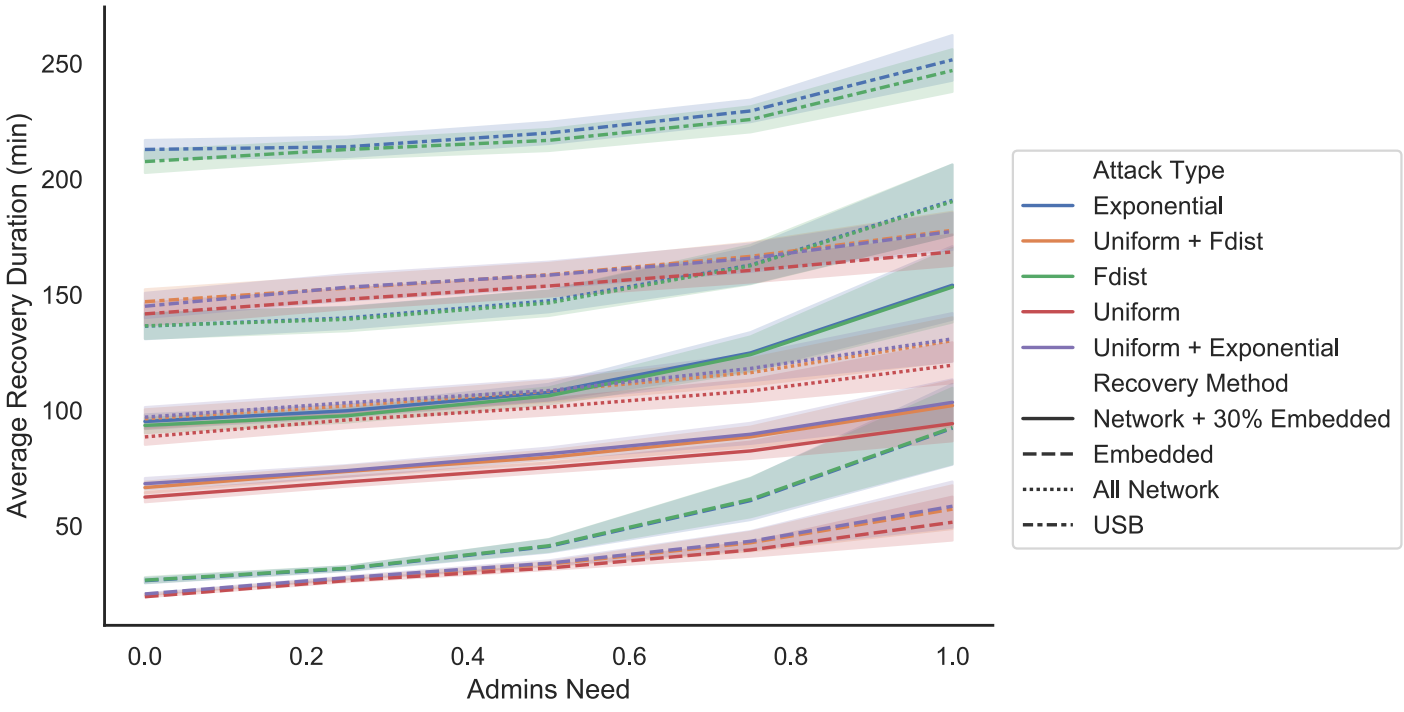



**Fig. 30.** Impact of Admins Need on average recovery duration

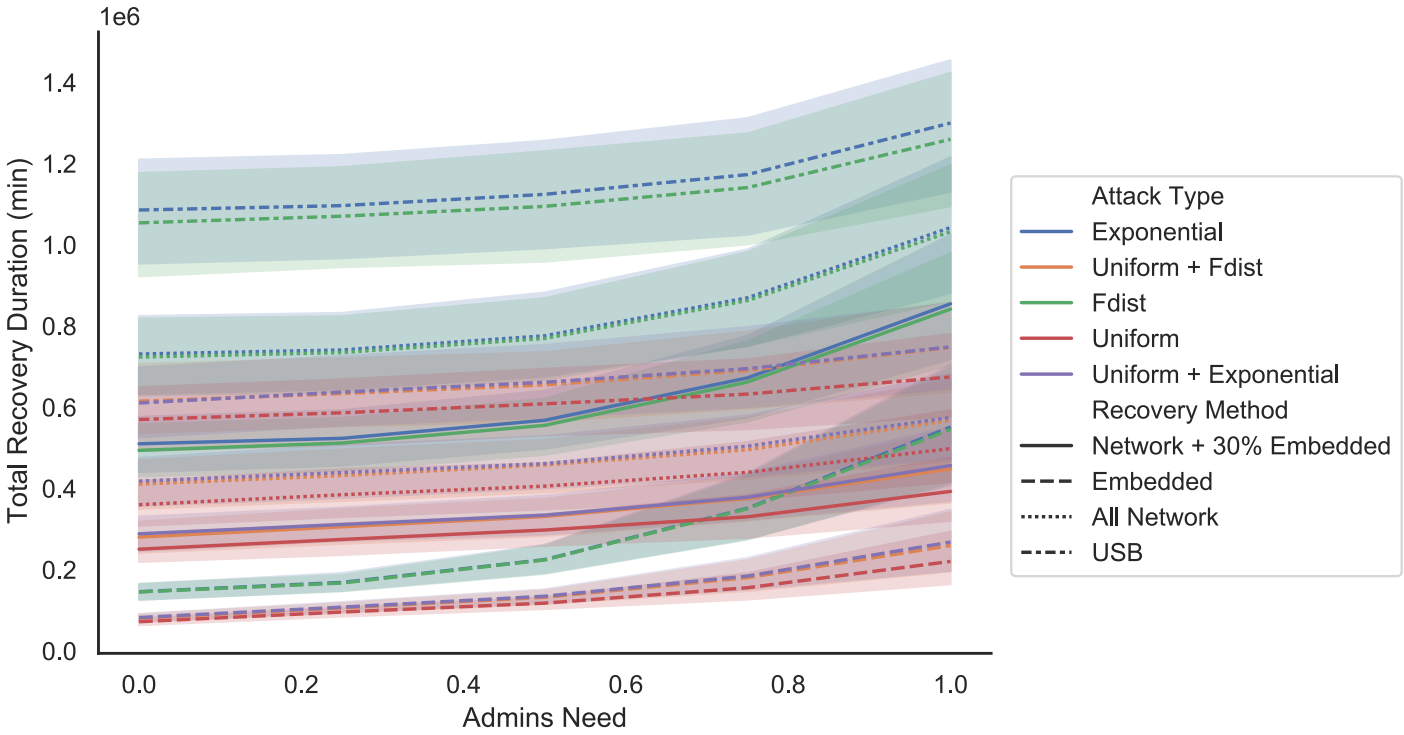



**Fig. 31.** Impact of Admins Need on total recovery duration

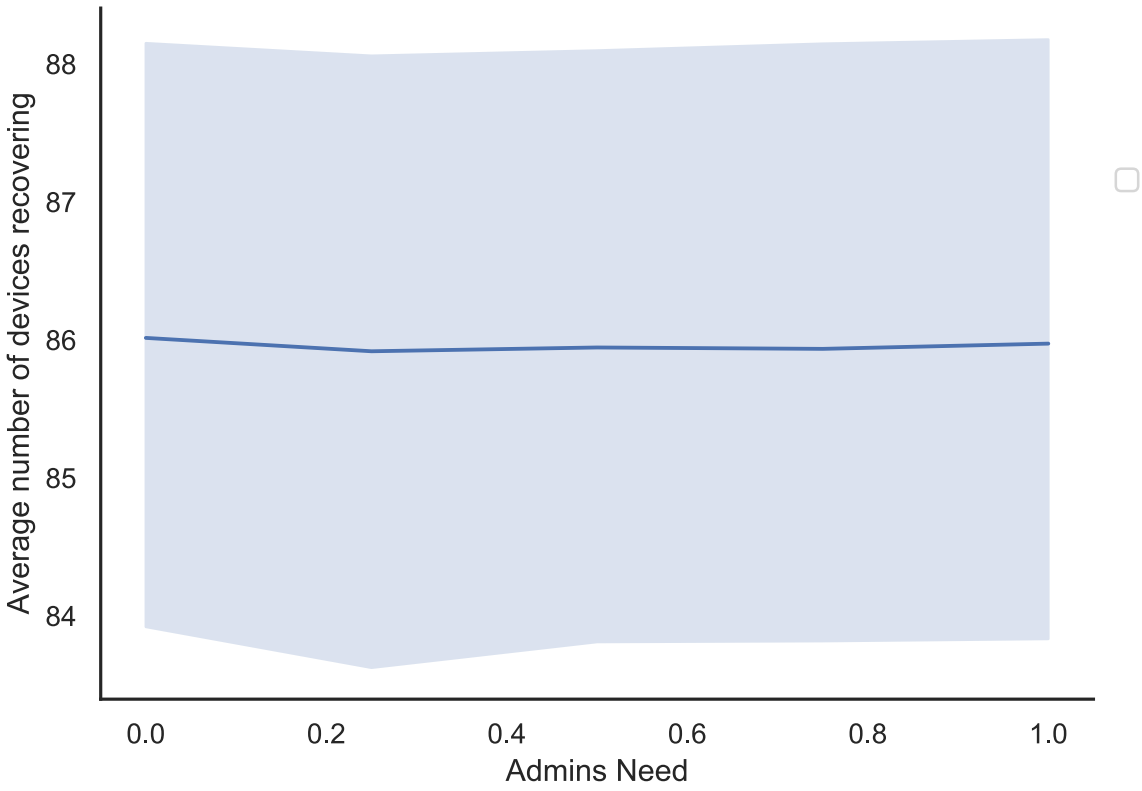



**Fig. 32.** Impact of Admins Need on the average number of devices recovering

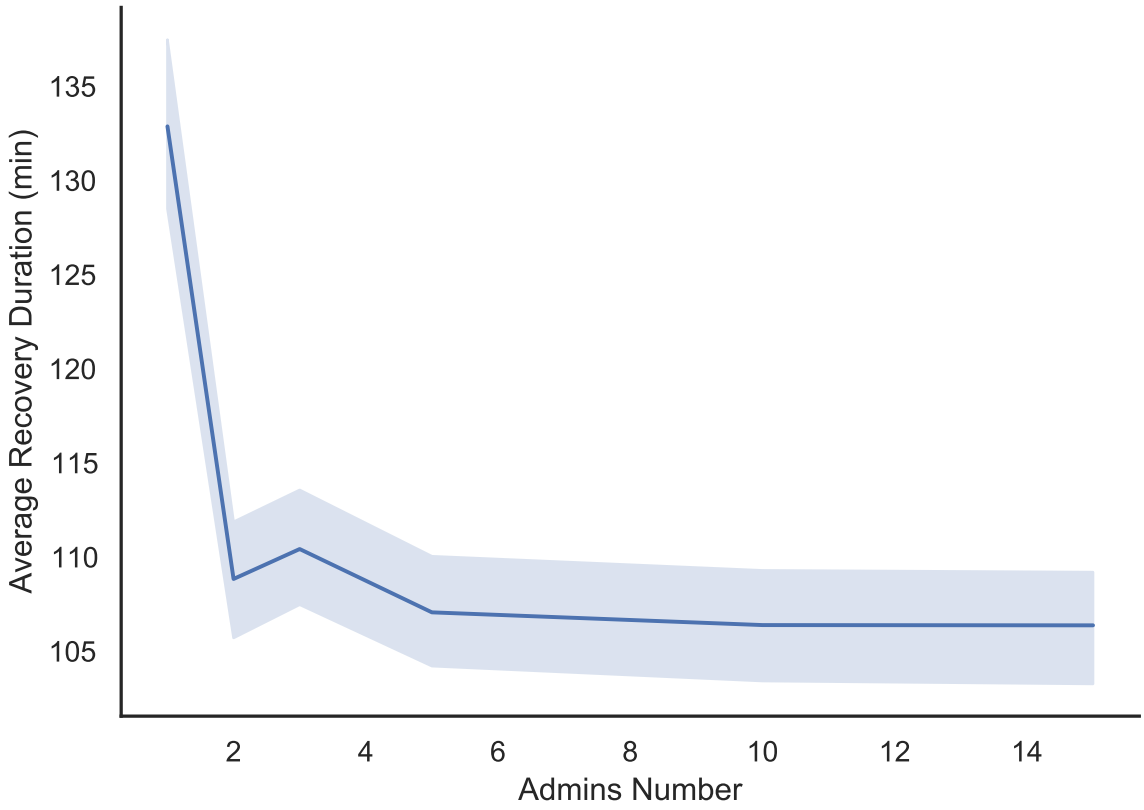



**Fig. 33.** Impact of Admins Number on average recovery duration

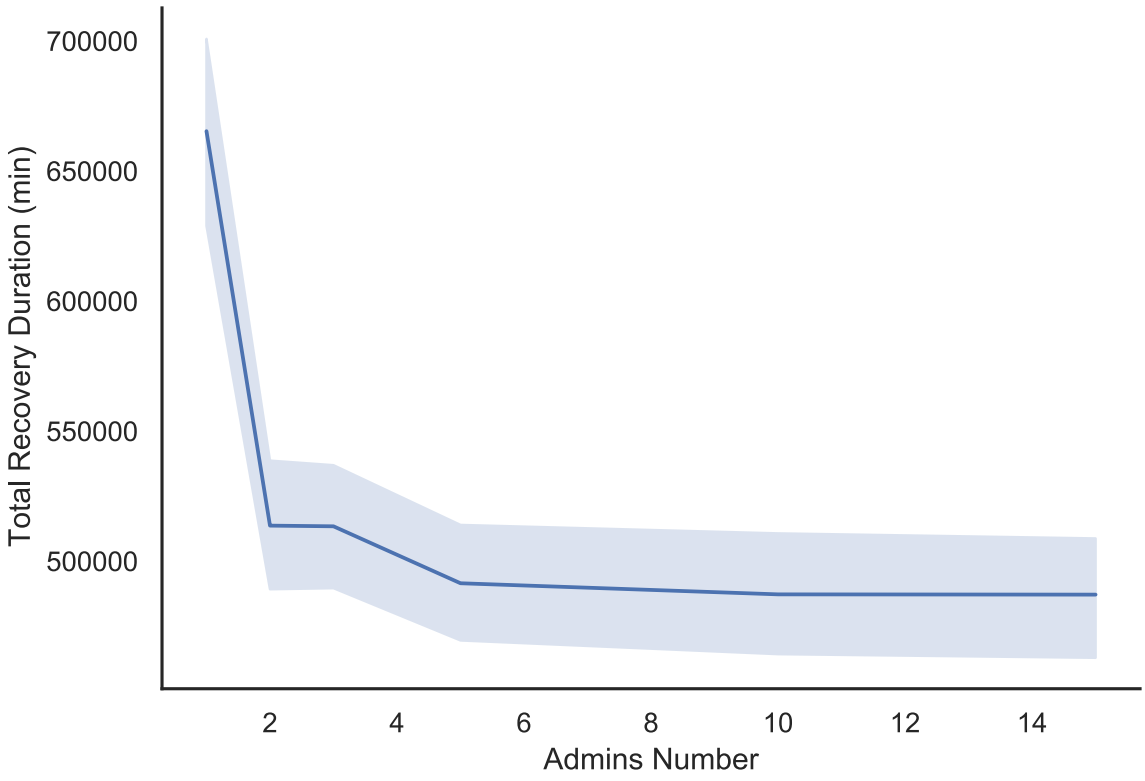



**Fig. 34.** Impact of Admins Number on total recovery duration

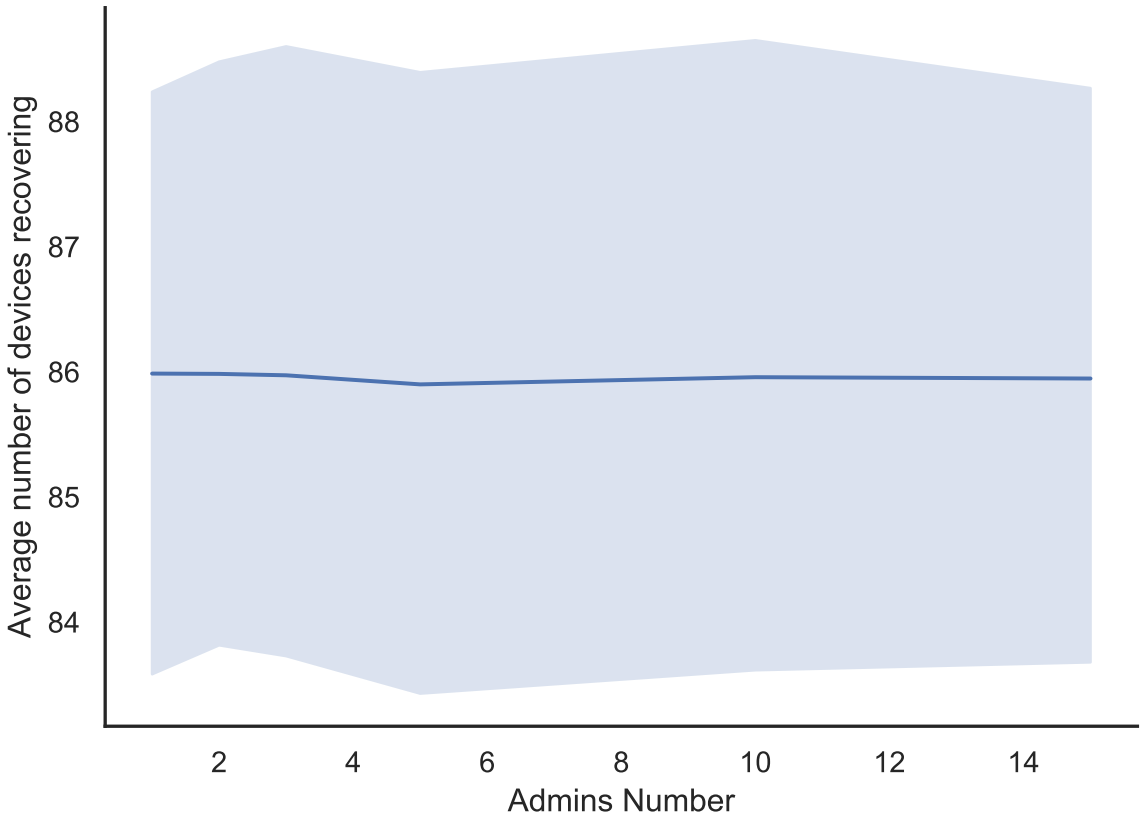



**Fig. 35.** Impact of Admins Number on average number of devices recovering

## Appendix D: Model Parameters

This section contains information regarding the array of parameters used in the initialization and execution of our models. Tables 5 and 6 contain manually gathered recovery timings for the network, embedded and usb recovery. The same timing values have been also used in (10). Furthermore, tables 7 and 8 describe in detail the types, values and meanings of both the static and variable parameters used in the model.

| Recovery Steps | Sure Recover | Embedded |
|---|---|---|
| Initialize Recovery | 40 | 30 |
| Copy from embedded | N/A | 20 |
| Download and verify Recovery Agent | 100 | N/A |
| Boot to recovery agent | 15 | 25 |
| Initialize Drive | 25 | N/A |
| Download Imaged | 1130 | N/A |
| Verify Image | 50 | 40 |
| Extract Image | 180 | 145 |
| Install Drivers | 60 | 80 |
| Windows Installer to Config Screen | 480 | 480 |

**Table 5.** HP Sure Recover times for both the network-based recovery and embedded recovery. Times are given in seconds and are based on a number of recovery cycles.

| USB Step | Time |
|---|---|
| Create Bootable USB | |
| Download recovery tool | 60 |
| Run Tool | 45 |
| Partition USB | 35 |
| Download Imaged | 1260 |
| Extract Image to USB | 3120 |
| Install from USB | |
| Boot USB to Installer | 120 |
| Partition Disk | 60 |
| Install Windows and Drivers to disk | 780 |
| Windows Installer to Config Screen | 480 |

**Table 6.** USB Based Recovery times. The first part of the table shows the steps in using a recovery tool to create a bootable windows installer. The second section shows times for the install from the USB stick. Again, times are quoted in seconds.

| Parameter Name | Type | Values | Meaning |
|---|---|---|---|
| **device_scenario** | Categorical | 1 - USB recovery<br>2 - Network recovery<br>3 - Embedded Recovery<br>4 - Network recovery with 30% embedded recovery | The type of recovery technique the devices will use. |
| **attack_scenario** | Categorical | 1 - Uniform<br>2 - Exponential<br>3 - Fdist<br>4 - Uniform + Exponential<br>5 - Uniform + Fdist | The types of distributions used when sampling for the timings of the attack/malware packets. |
| **infection_probability** | Float | [0.1, 0.3, 0.5, 0.7, 0.95] | The probability a device will get infected when hit by a malware packet. |
| **attack_duration** | Int | 1 - 2 hours<br>2 - 4 hours<br>3 - 8 hours | The duration in hours that the malware attack will take. |
| **admins_nr** | Int | [1, 2, 3, 5, 10, 15] | The number of admins that can be found in a big office. |
| **admin_need** | Float | [0.0, 0.25, 0.50, 0.75, 1.0] | The probability a user needs a physical admin, or to speak with an admin remotely to start the recovery process. |

**Table 7.** Model Variable Parameters

| Parameter Name | Type | Values | Meaning |
|---|---|---|---|
| **num_iterations** | Int | 50 | The number of times the model will be run with the same parameter configuration. |
| **proc_num** | Int | 50 | The process number running a certain model iteration. Used for running multiple iterations in parallel. |
| **nr_of_samples** | Float | 300 | The number of attack packets sent by the malware model in a specified attack_duration. |
| **attack_targets** | [String] | ["Office 1 LAN", "Office 2 LAN", "Travel WiFi", "Small Office 1 LAN", "Small Office 2 LAN", "Office 1 WiF", "Office 2 WiFi", "Small Office 1 WiFi", "Coffee WiFi","Home WiFi","Small Office 2 WiFi"] | The names of the locations where devices to be targeted by malware can be found. |
| **phisical_admin_time** | Float | 15m | The time it takes for an admin to physically perform the needed operations to start a recovery process. |
| **admin_movement_time** | Float | 120m | The time it takes for an admin to physically move to a location where a recovery process is needed. |
| **admin_remote_time** | Float | 20m | The time it takes for an admin to guide a user remotely to start a recovery process.. |
| **os_images** | [OSImage] | [windows10iso, windows10wim, recovery_agent] | The list of available recovery images on the recovery server. |
| **max_office_devices** | Int | 30 | The maximum number of devices that can connect to the network from an office. |
| **max_home_devices** | Int | 65 | The maximum number of devices that can connect to the network from home. |
| **max_coffee_devices** | Int | 65 | The maximum number of devices that can connect to the network from a coffee shop. |
| **max_travel_devices** | Int | 30 | The maximum number of devices available for travelling. |
| **scale_uni** | Float | [72, 144, 288] | Scaling factor used for interval boundaries movement for attack packets that do not use mixed distributions. |
| **scale_dst** | Float | [400, 750, 1500] | Scaling factor used for interval boundaries movement for attack packets that use mixed distributions. |
| **num_office_desktops** | Int | 20 | The actual number of desktops in a big office. |
| **num_office_laptops** | Int | 20 | The actual number of laptops in a big office. |
| **num_small_office_desktops** | Int | 10 | The actual number of desktops in a small office. |
| **num_small_office_laptops** | Int | 10 | The actual number of desktops in a small office. |
| **num_travel_laptops** | Int | 5 | The actual number of laptops that can be used for travelling from a big office. |
| **server_speed** | Float | 10.0 * 1024^4 | The upload/download speed of the recovery server. |
| **office_lan_speed** | Float | 1.0 * 1024^3 | LAN upload/download speed for a big office. |
| **office_wifi_speed** | Float | 150.0 * 1024^2 | Wifi upload/download speed for a big office. |
| **office_link_speed** | Float | 1.0 * 1024^3 | Switch upload/download speed for a big office. |
| **small_office_lan_speed** | Float | 1.0 * 1024^3 | LAN upload/download speed for a small office. |
| **small_office_wifi_speed** | Float | 150.0 * 1024^2 | Wifi upload/download speed for a small office. |
| **small_office_link_speed** | Float | 200.0 * 1024^2 | Switch upload/download speed for a small office. |
| **coffeeshop_download_speed** | Float | 12.0 * 1024^2 | Download speed for a coffee shop. |
| **coffeeshop_upload_speed** | Float | 2.0 * 1024^2 | Upload speed for a coffee shop. |
| **coffeeshop_link_speed** | Float | 100.0 * 1024^4 | Switch upload/download speed for a coffee shop. |
| **home_download_speed** | Float | 50.0 * 1024^2 | Download speed for home. |
| **home_upload_speed** | Float | 4.0 * 1024^2 | Upload speed for home. |
| **home_link_speed** | Float | 100.0 * 1024^4 | Switch upload/download speed for home. |
| **travel_download_speed** | Float | 30.0 * 1024^2 | Download speed for a travelling location. |
| **travel_upload_speed** | Float | 5.0 * 1024^2 | Upload speed for a travelling location. |
| **travel_link_speed** | Float | 100.0 * 1024^4 | Switch upload/download speed for a travelling location. |
| **office1_usb_images** | [OSImage] | [windows10iso] | The recovery images available on usb in office1. |
| **office1_usb_blanks** | Int | 7 | The number of blank usbs in office1. |
| **office2_usb_images** | [OSImage] | [windows10iso] | The recovery images available on usb in office2. |
| **office2_usb_blanks** | Int | 7 | The number of blank usbs in office2. |
| **small_office1_usb_images** | [OSImage] | Empty | The recovery images available on usb in small office1. |
| **small_office1_usb_blanks** | Int | 3 | The number of blank usbs in small office1. |
| **small_office2_usb_images** | [OSImage] | Empty | The recovery images available on usb in small office2. |
| **small_office2_usb_blanks** | Int | 3 | The number of blank usbs in small office2. |
| **home_usb_images** | [OSImage] | Empty | The recovery images available on usb at home. |
| **home_usb_blanks** | Int | 0 | The number of blank usbs at home. |
| **coffee_usb_images** | [OSImage] | Empty | The recovery images available on usb in a coffee shop. |
| **coffee_usb_blanks** | Int | 0 | The number of blank usbs in a coffee shop. |
| **travel_usb_images** | [OSImage] | Empty | The recovery images available on usb in a travelling location. |
| **travel_usb_blanks** | Int | 0 | The number of blank usbs in a travelling location. |
| **travel_movement** | [Movement] | [Movement("Travel", "Travel WiFi", Uniform(4hours, 5days)), Movement("Office 1", "Office 1 WiFi", Uniform(2hours,6hours)), Movement("Office 2", "Office 2 WiFi", Uniform(2hours,6hours)), Movement("Small Office 1", "Small Office 1 WiFi", Uniform(2hours,6hours)), Movement("Small Office 2", "Small Office 2 WiFi", Uniform(2hours,6hours))] | The list of possible movements from a travel location. |
| **large_office_movement** | [Movement] | [Movement("Home", "Home WiFi", Uniform(0hours, 3hours)), Movement("Office ", "Office WiFi", Uniform(5hours, 8hours)), Movement("Coffee Shop", "Coffee WiFi", Uniform(20minutes, 2hours))] | The list of possible movements from a large office location. |
| **small_office_movement** | [Movement] | [Movement("Home", "Home WiFi", Uniform(0hours, 3hours)), Movement("Small Office ", "Small Office WiFi", Uniform(5hours, 8hours)), Movement("Coffee Shop", "Coffee WiFi", Uniform(20minutes, 2hours))] | The list of possible movements from a small office location. |

**Table 8.** Model Static Parameters