Modelling Organizational Recovery

Adrian Baldwin[1], Tristan Caulfield[2(✉)], Marius-Constantin Ilau[2], and David Pym[2,3]

[1] HP Labs, Bristol, UK
[2] University College London, London, UK
{t.caulfield,marius-constantin.ilau.18,d.pym}@ucl.ac.uk
[3] Institute of Philosophy, University of London, London, UK

Abstract. Organizations today face a significant set of sophisticated information security threats, including rapidly spreading malware that can affect many devices across the organization. The impacts of such attacks are amplified by customers' rising expectations of high-quality and rapid delivery of products and services, as well as by organizational attempts to increase demand artificially. This leads to the development of defence mechanisms that prioritize availability and integrity for the sake of reducing the overall time of organizational recovery. However, such mechanisms and strategies around recovery must suit the organization that deploys them. Each organization will have different priorities in terms of budget, speed of recovery, and priority of services or devices, and all of these will be impacted by the architecture of the organization and its networks. In this paper, we show how modelling can play a role in helping organizations understand the consequences of the different recovery mechanisms and strategies available to them. We describe a rigorous modelling framework and methodology grounded in mathematical systems modelling and simulation, and present as an example a comparative analysis of recovery strategies and mechanisms on a medium-scale organization.

Keywords: Distributed systems · Malware · Modelling · Recovery · Compositionality · Interfaces · Organizations

1 Introduction

The problem of maintaining the integrity of data and availability of services of an organization represents, in today's transactional economy, an issue at least as important as the optimization of internal operations for the reduction of cost.

This work has been partially supported by the UK EPSRC research grant EP/R006865/1.

D. Jiang and H. Song (Eds.): SIMUtools 2021, LNICST 424, pp. 284–314, 2022.
https://doi.org/10.1007/978-3-030-97124-3_23

This can arise, for example, as a side-effect of sales strategy, focussed on producing short-term bursts of extremely high-volume of sales by artificially increasing customer demand around so-called 'sales events', such as Black Friday, Cyber Monday, and others. Even though not every organization is cost-oriented, the same concept of high operational demand in a short period of time can be understood according to the business' primary goal. For example, an online retailer during a sales event, a hospital during a pandemic, or a border control force the day after significant legislation has been introduced all may face possibly existential threats when confronted with poor service or even loss of service availability when most needed.

In such cases, the organizational loss increases because of a trust violation between the supply and demand and additional reputational 'recovery' measures may be needed, in the sense described in [13]. However, this impact varies across industries: in their comparative study of stock performance post cyberbreach, Tweneboah et al. [47] show that the healthcare industry in particular lacks resilience, the technology industry is somewhat neutral, in that the stock price remains fairly stable, and the financial industry actively manages to increase stock value, even post-breach.

In this context, the security professional can interpret such events as single points of failure for the business: the impact of a loss of availability of services during such an event is significantly higher than during business-as-usual times. This motivates the organizational need for mechanisms for quickly restoring service availability to a state in which data integrity is maintained. Such mechanisms are not limited to defensive purposes. They can provide valuable time optimizations in non adversarial contexts, such as recovery needed because of 'WinRot' [17] phenomenona—registry corruption, mismanagement of DLL libraries, or disk fragmentation—or scheduled patching, where the same mechanism used for recovery is used to deploy pre-updated system images.

Threats to availability and integrity come in many forms: malware, denial of service, faulty patch updates, disgruntled employees, database query mistakes, and so on. Summarizing, they arise from both malicious and non-malicious actions, internal or external. Active defense against such threats is possible, but dependent not only on technical measures and controls, but also on policy compliance and employee behaviour. Therefore, particularly in situations where the loss of business continuity even for small time periods can lead to acute organizational damage, the risk of employing solely active countermeasures against possible attacks is not enough. Following the defence in depth principle, we focus on a practical second-line control mechanism that can tackle both inside and outside threats with respect to availability and integrity: a recovery mechanism. The development of such mechanisms marks a shift from traditional defence strategy that focuses on practically countering attacks to economic-based deterrence that seeks to reduce the impact of attacks until they eventually become economically infeasible.

As might be expected, multiple approaches to recovery exist and a subset of interest will briefly be outlined and explored in Sect. 2. However, their positive

impact on a company depends on the nature of the attack tackled, spread, size, recovery time, cost of deployment, existing policies, and employee knowledge and behaviour. Therefore, the problem of recovery in the context of a heterogeneous organization is complex, but particularly suitable for comparative modelling of solutions.

Relevant examples of how a simulation-based methodology has been used to assess and improve organizational supply chains can be found in [15] and [23]. Furthermore, analytical approaches can be found with respect to modelling the impact of cyber-attacks on web sales. For example [16], clearly shows the impact of lengthy cyber-attacks on organizational costs, and although the authors do not suggest it, a recovery solution would serve as a good mitigation in the described scenario.

In this paper, our primary aim is to demonstrate how a modelling and simulation approach, based on rigorous mathematical foundations, can be used to help organizations understand the problem and the consequences of different recovery choices for their organization, and help them make better decisions about this challenging problem. 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 different offices, the devices used, the value of the information on different devices, and the network structure will all vary between organizations. A compositional modelling approach provides the flexibility required to create models that can be adapted to capture all of these differences between organizations. We will demonstrate this approach by using examples to explore the relationship between the impact of different deployment strategies under varying cyber-attack conditions and the recovery time in different scenarios, and discuss how the models can be adapted to different organizations, and extended to cover different scenarios and recovery strategies.

In Sect. 2, we describe some of the practical implications that modern destructive malware techniques have brought into the organizational recovery space and list some of the recovery strategies and mechanisms that organizations can choose from when attempting to manage this risk. Section 3 focuses on explaining the basic functional and methodological concepts behind our modelling approach, with references to the relevant literature that further develops these concepts. Building on top of Sect. 3, Sect. 4 describes the conceptual model of organizational recovery. Section 5 explains the discrete event simulation models of the recovery scenarios including the parameter space and choices, the results obtained, and validation. Lastly, Section 6 attempts to offer some organizational guidelines based on the results of the prior section while also illustrating some of the further work considerations that could be pursued in the future.

A systematic comparison of our modelling approach with other approaches is beyond the scope of this short, introductory paper. Such a comparative study is deferred to another occasion.

2 The Recovery Problem

In June 2017, the NotPetya ransomware [18] spread across the world and caused considerable disruption for enterprises because of its destructive capabilities. For example, Maersk's IT systems, including 49000 laptops and 3500 servers, were destroyed—effectively wiping out its ability to operate. The malware was introduced to the world through an update to an accounts package, but included the Eternal Blue library (also used in WannaCry [25]), which used vulnerabilities in the SMBv1 protocol to spread quickly between connected computers, particularly within enterprises. NotPetya not only encrypted files on the victims computer, but also overwrote the master boot record [43], hence stopping Windows from booting. More recently, human operated ransomware campaigns have been becoming an increasing threat to businesses [29,46]. Here, after an initial infection, the ransomware spreads using a variety of lateral movement techniques, usually based on acquiring accessed to privileged accounts and using system management tools such as WMI or powershell to spread throughout the enterprise. Such attacks often leave backdoors in infected systems, making reimaging necessary. Such a process can take a well-prepared organization weeks to recover from—and months for larger organizations or those without good backups and IT support. The details of the process and how organizations often end up paying ransoms rather than rebuilding their systems are described in [48].

It is not just ransomware that spreads over networks necessitating recovery at scale. Malware families, such as Emotet [10], also have mechanisms to spread rapidly through email. Spreading patterns have evolved to include WiFi [5], 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 [24], with recent surveys showing incident response processes often leading to re-imaging [8].

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 [14]. 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, showing how modelling and simulation can be used to aid the decision-makers in the choices they make.

2.1 Image Management and Recovery

There are several approaches that companies use in order to maintain and re-image 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. Taking full backups is becoming less common 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 [42]. Microsoft provide a management deployment toolkit [31] 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 [30]. 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 [28]. 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 [32], 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 used 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 [27]—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. Thus, 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' [21]) 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.

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. 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.
- 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. At the same time, the enterprise will need to plan for remote workers either working from home or as they travel.
- 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.

3 The Modelling Technique

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 tasks—that is, deliver services—and communicate by sending information or messages over network connections.

While the origin of this view is very specific to its focus on computer systems, its concepts can be taken more generally to provide a useful metaphor for understanding all types of systems and, finally, ecosystems. The models we use in this paper are executable models, implemented in Julia code, and constructed using a modelling framework designed around this distributed systems perspective [7]. The framework itself is built on a rigorous mathematical foundation—see, for example, [1,11,12,39]—which gives an understanding of the components and operation of distributed systems. 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). Mathematically, we require a topological structure that supports the concept of connected places. In many examples, found widely in the complex systems modelling literature, *directed graphs* are quite sufficient. Generalized notions of graph, as well as more exotic topological structures, can also be considered. The implementation uses a directed graph structure: locations have links to other locations, which constrain how resources may be moved from one location to another.
- *Resource*—Resources exist at locations and can move between them according to the locations' connections. In the general view, they can represent anything, including physical objects, people, and information.
  Mathematically, we start from the basic observation that resource elements can be combined and compared. This leads us to consider *ordered monoids*—and collections thereof—of resource elements. In some circumstances, slightly more complex structures may be required [1,11,12]. Examples of such structures include $(\mathbb{N}, +, 0, \leq)$—that is, the natural numbers—and computer memory cells, combined by concatenation and ordered simply by equality, as used in Separation Logic [22,41] and its many variants.
  Resources are implemented as instances of objects. Different types of resource are defined by declaring different types in the code. Each location has a collection of resources that are present at that location.
- *Process*—Processes execute relative to resources at locations and manipulate resources as they do so.
  Mathematically, we work employ a calculus of processes that is closely related to Milner's *Synchronous Calculus of Communicating Systems (SCCS)* [33,34]. The implementation of processes in code is based around a discrete event scheduler. Processes are written as functions whose execution can be interrupted by `hold` statements, which pause a process for some specified amount of (simulation) time, and `claim` statements, which pause a process until specified resources become available at specified locations.
  The first statement, `hold`, allows processes to model the passage of time—a real-world process that takes some amount of real-world time can be modelled

by a model process using `hold` statements to simulate the passage of time.
The second statement, `claim`, is how processes manipulate resources and how processes interact with each other. Once a process has claimed a resource it can move it from location to location, or remove it. When a process has finished manipulating a resource, it should `release` it. Only one process can claim a resource at any given time. If a process tries to claim resources that are not present (or are claimed by other processes) in the specified locations, it will pause until the resources are released.
It is easy to see how complex models can be built using these simple constructs. For example, a process that represents a door opening can't continue until the resource representing the key is present; or a process of a file download might move a resource from a server location to a client location, after pausing for the length of time the download takes; or, if two processes are representing cars trying to park, only one will be able to claim the resource that signals a particular parking space is free. This shows how processes communicate and coordinate.

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. We think of this in terms of events that happen at the boundary of the model and cause an action to occur in the model itself.
Mathematically, we capture the incidence of events at the boundary of a model—be they inbound, originating outwith the model, or outbound, originating within the model, using probability distributions. For example, an arrival rate for requests to a server may have a negative exponential distribution.
The concept of environment is implemented in code as a collection of environment processes—distinct from the processes in the model itself—which create resources or start other processes in the actual model. These can incorporate probability distributions to model the delay between events.

Overall, models are based on a transition system with evolutions of states—with location L, resources R, and process, E—the form

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

read as 'the process E executing with access to resources R at location L evolves by the action a to be the process E' executing with access to resources R' at location L'.

Such evolutions are defined by an operational semantics that is structured by the process combinators described above. Equality between states L, R, E and L', R', E', denoted $L, R, E \sim L', R', E'$, is given by bisimulation equivalence [1,11,12]. We sometimes denote states by $\mathcal{S}, \mathcal{T}$, and so on.

Along with the transition system comes a logic of states. In logical terms, states L, R, E form the worlds, in the sense of Kripke models, of a model of language of propositional assertions—see, for example, [1,11,12,19,19,33,34,44]. We write

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

to denote that the property ϕ is true of the process E executing with resources R at location L. In modelling ecosystems, such logical properties can be used to specify the properties that required of the interfaces of two models in order for the models to be composable.

This generalization of the idea of a distributed system, together with its mathematical tools, 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 [6], where a programmatic description of the system (in terms of locations, resources, and processes) is run to simulate the behaviour of the system [7,9,11,45].

An early implementation of these ideas, Gnosis [11], has been used in significant commercial applications [2–4] derived from an industry-based research project [20]. The current framework, the Julia `SysModels` package ([7]) is an improved, more modern implementation of these ideas that includes new capabilities such as composition of models.

The ability to compose models is important for modelling larger systems and ecosystems. During modelling, these systems can be decomposed into smaller parts, which can be modelled separately and then recombined, and which helps manage complexity. How does composition work in the distributed systems approach?

We start with 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 naturally support the concept of local reasoning. Such an account of reasoning requires a mathematical conception of the distributed systems metaphor on top of which interfaces and composition can be defined.

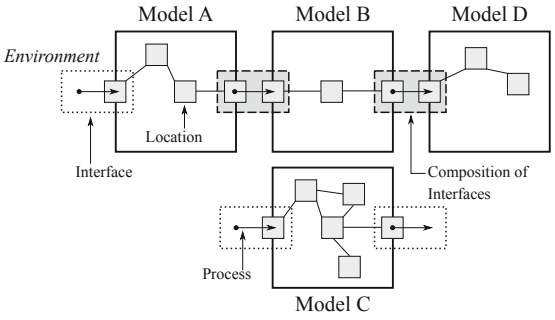

Fig. 1. Interfaces, composition, and substitution

We can identify here a local reasoning principle, or *frame rule* [22,37–39,49]. We begin by setting up some notation for the states of the various component models depicted in Fig. 2. The details of this set up may be found in [45]. 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' that part of the model N_i that is not coincident with the interface 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.

Milner [35,36] 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. Our notion of interface is also related to Lynch and Tuttle's input/output automata [26].

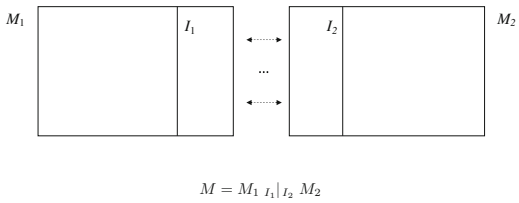

Fig. 2. Local reasoning and the frame property

At the end of Sect. 4, after we have explained the set-up of the specific model we use in this paper, we discuss briefly its compositional structure and how this would facilitate local reasoning about the components of the model.

4 The Model

We wish to build a model that can help organizations understand the consequences of different device recovery technology and strategy choices in the presence of various kinds of attacks. Different organizations will be of different sizes, have different structures, use different technologies, place different values on their information and system assets, and have different understandings of what a successful recovery looks like. Because we want this modelling approach to be useful to a wide range of organizations, we focus on constructing a generic model that captures the systems, structures, and behaviours necessary for thinking about organizational recovery, and which can be parametrized and configured to create a model of a chosen organization.

Figure 3 illustrates the high-level architecture of our model, in the configuration used for this paper. It consists of three models which are composed together: a device model, a network model, and a server model. Each of these models can be configured differently to represent different organizations, and, because of the compositional structure, additional models could be substituted or added if the current models do not capture the necessary details of a particular system.

Device Model—The device model is the most complex model of the three. It is designed to be able to represent the devices (essentially, computers in the form of laptops and desktops) used by an organization, the locations these devices may be in (such as offices, homes, hotels, and so on), the movement of devices between these locations, and the different ways devices at each of these locations can recover.

Network Model—This models computer networks over which information can be sent. It is configurable; different network topologies can be constructed, with different bandwidths available on different segments. When segments are overloaded, transfers through them are throttled.

Server Model—This is the simplest of the three models. The server model listens for requests for operating system images sent by devices and responds by transmitting the OS image to the requesting device back across the network.

4.1 Operation

We describe the operation of the models in terms of locations, resources, and processes.

In the device model, locations are used to represent two things: the physical locations where devices may be found—offices, homes, etc.—and the network endpoints where information can be sent and received.

Resources are used to represent devices, which can be moved around the physical locations, and things used in the various recovery processes: USB sticks, OS images, and image requests and responses which are sent and received over the network. Resources are also used to represent the *availability* of network endpoints. 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 of the device's behaviour, 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), a method of recovery to use, whether or not the device should recover, and, if so, at what time.

These last two can be varied to model different intensities of attacks. More devices recovering in a shorter time period models a more intense attack—for example, faster-spreading malware. Fewer devices, possibly over a longer period of time models a less intense attack.

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, the device process initiates this at the appropriate time.

In this paper, we look at three recovery methods. USB recovery, where devices install a fresh OS from a USB stick; network recovery, where devices request and receive an OS image over the network from an image server; and, embedded recovery, where devices have a built-in storage capability that is used to hold an OS image for recover.

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. For network recovery, the process has a few more steps. It 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; 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 USB recovery, 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. Throughout all these steps in the process are time delays modelling the length of time it takes to, for example, verify an image after download, or copy an image to disk, or run the installer.

In the network model, locations are used to represent each of the network endpoints with one additional location representing data resources in transit. This 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 most simple. It has a network endpoint location, and a process which waits for requests to arrive from the network and sends responses back.

4.2 Composition

These three models compose together to form a model of device recovery in which OS images can be obtained from an image server over a network. As discussed in Sect. 3, composition of models occurs at interfaces. In this case, interfaces are defined at each network endpoint. The network model becomes the glue that sticks the server model and the device model together. The server 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.

The compositional approach to the design of these models facilitates the ability to reason locally about the underlying components, providing two primary advantages: modularity, for further extension, and the ability to focus the analysis on a singular model component without the need to reason about its relationships with other components.

These ideas are important for future extensions to this modelling approach. The network model does not know anything about the type of data resources being transmitted over it; it only knows that when it receives a resource at one interface, it should deliver it to another. Additional models could easily be composed with the network model, too. For example, another server model could be used to store data backups received from devices, or a model of attacks, that transmits resources representing malware to devices. These examples and the existing device and server models work when composed without knowing how the other models function because of the well-defined interfaces between them. Knowing the specification of the interface between the models allows reasoning—that is, *local reasoning*, as explained in Sect. 3—about the components of these

models in a similar fashion, without the fear of the arguments being invalidated by specific cases of interaction between the two models.

5 Scenarios, Parameters, and Validation

We describe a given recovery scenario and show how the model can be used to reason about different recovery choices including where to place recovery images and what recovery mechanisms, or mix of mechanisms, to support. The model currently covers the generic aspects of the recovery mechanisms, such as network bandwidth and other resource contention, but we discuss—informally for now—how local reasoning in the model can be used to model specific enterprise IT processes to produce more tailored results.

Here, we explore attack scenarios where fast spreading malware hits the enterprise to explore how such wide scale attacks will stress the enterprises recovery processes. From the model we measure two aspects, recoverability and the time to recover and these need to be placed along side deployment, support costs and security as the enterprise makes deployment choices. The scenarios and configurations we present here are examples to illustrate the modelling approach; when deploying this approach in a real organization, a different range of scenarios and recovery mechanisms may be required.

5.1 Recovery Mechanisms

Section 2 described the broader recovery space. Here, we consider a subset of these enterprise choices and address one of the first questions that an enterprise will need to tackle: what mechanisms should be used for re-imaging? We compare the use of USB sticks with different options of using the bare-metal recovery (network and embedded) provided by HP 'Sure Recover'. These basic recovery mechanisms cover the way in which a fresh Windows image gets onto the disk. We have included timings up to the point at which the new image has booted and the OOBE starts, and the user configures the system. In our scenario, this is the same process and timings in each recovery scenario.

USB Stick. USB stick recovery can be used when the OS is no longer trusted and would typically be used in an enterprise that manages its own bespoke Windows images. Here, we consider the USB sticks with and without images as a resource that can become limited as the number of devices in a given location need to recover. The process can be somewhat more complex as good security practice means disabling USB boot and only re-enabling it through the BIOS on recovery. Where companies follow good practice and lock down the BIOS settings with passwords or other mechanisms, this can create support requirements as administrators will need to perform the re-imaging tasks. This is a two stage recovery process.

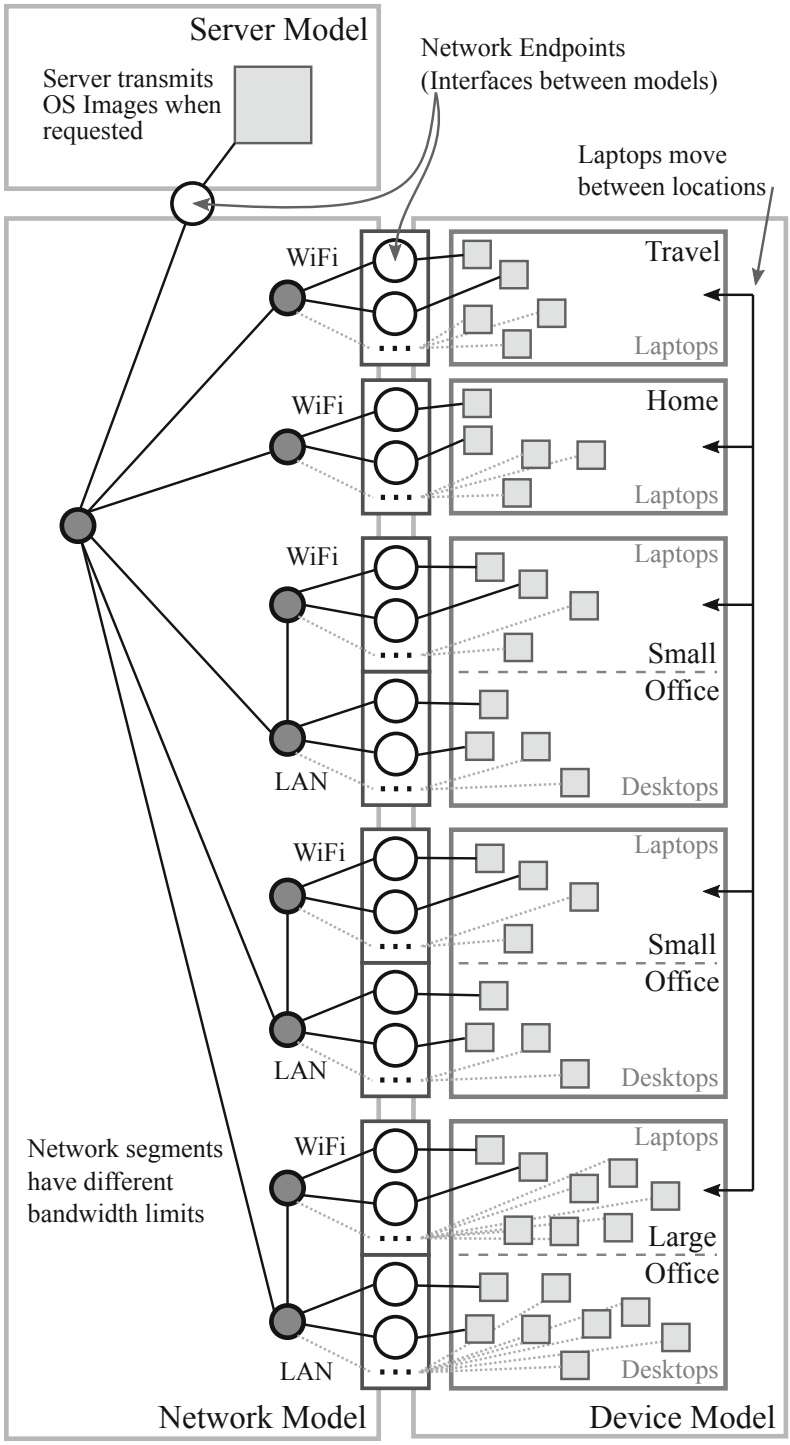

Fig. 3. Model of organizational recovery

1. The first stage involves preparing a recovery USB stick. For a standard Windows machine, the Windows recovery allows the creation of recovery media. However, in a company with its own managed image, the process will involve downloading an image and placing it on the USB stick. Typically, IT support will maintain a set of images ready for the inevitable re-imaging they expect to perform. But such resources will be to support day-to-day operations and resources may not be available in smaller offices with no on-site support.
2. The second part of the process is the re-imaging itself, where the USB stick is used to boot the PC. This requires triggering through the BIOS interface. The re-imaging process can be more complex for a user who requires disk partitioning.

In Table 2, we list the stages of recovery, along with some typical timings generated by following these steps a few times, in a home environment with a USB stick that is typically available in an office.

These times assume that a USB stick is readily available and that it can be reformatted without the need to save data elsewhere. It may often be the case that a user either does not have access to a USB stick or does not have access to a second working (or trustable) computer that they can used to download the image and format the USB stick. Within the model, we record failures when the user is not in an office with readily available resources.

HP 'Sure Recover'. An alternative to USB-based recovery is where the recovery process is built into the BIOS and where either the image is stored locally or fetched over the network. An example of this is HP 'Sure Recover', where either a user can trigger recovery through a BIOS option or it will be offered when a system will not boot (e.g., after an attack with malware such as NotPetya).

The basic HP 'Sure Recover' process first downloads a recovery agent, which initiates the recovery process, partitions the disk, and downloads the new full OS image over the network. Once downloaded, the image's signature is checked, it is written to the disk, and the Windows install process starts. Table 1 shows recovery times (from the same home environment as for the USB stick timings). The times given in the model are based on the size of a windows image being downloaded through the network model, taking account of network contention rather than from sample measurements. We refer to this as network recovery. Embedded recovery in our model takes advantage of local storage for the image and network downloads are not required.

5.2 Enterprise Scenario

The model described in Sect. 4 allows an enterprise's endpoint set-up to be described in terms of the devices, their locations, and any movement between locations. This allows the enterprise to set up the model in a way that represents its organization and how staff move between offices and other locations such as working at home or in coffee shops. Given this set-up we can then specify different recovery mechanism and vary parameters such as the location of image

servers, the numbers of available USB sticks, and the network speeds in offices. Finally, we can set up a variety of failure scenarios, such as rapidly spreading destructive malware or just random infections and failures, that drive the need for recovery. 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 analysis, to aid decision-making.

We create a basic scenario, as depicted in Fig. 3, in which we have a large office with 40 desktops (systems that typically stay in the office) and 40 mobile workers who have movement profiles taking laptops between home, the office, and cafés (which represent other non-office locations). Within this office, we assume some level of IT support and when using USB based recovery the availability of some USB sticks with current images as well as a number of spare USB sticks. The large office is also well connected to the internet and has the possibility of having servers to host recovery images. We then have two smaller offices, each having slower internet connectivity and fewer users (10 fixed (desktop) and 10 mobile (laptops)). We assume minimal IT support and, in the case of USB stick recovery, that we do not have prepared images (but USB sticks are available).

In each of the offices, the mobile workers spend most of their day in the office, but will also move to other locations such as the home and cafés. We have a group of 20 travelling workers who move between a variety of networks such as airports and hotels, as well as appearing in the office. The public networks (airport, cafe, hotel) are assumed to be quite slow, with the homes having good home broadband speeds. In addition, we assume that as staff are mobile or working at home they do not have ready access to spare USB sticks.

We look at four recovery approaches.

- *USB.* USB stick recovery with the availability of USB sticks with and without images as set out above.
- *Network.* Network only based recovery for example using HP's 'Sure Recover'
- *Embedded.* A mix of laptops (mobile workers) HP's 'Sure Recover Embedded' (with images stored on devices) and the desktops just having HP's network based 'Sure Recover'.
- *30% Embedded.* 30% of the office workers laptops and all travelling staff having HP's 'Sure Recover' Embedded with the rest relying on HP's 'Sure Recover' for network-based recovery.

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.

We add in an additional scenario to demonstrate the advantages that can come with planning where network images are located and, in this case, add an image server into the large office.

We test recovery under quite extreme attack scenarios assuming rapidly spreading destructive malware. We have spread-rates running a spread over a 2, 6, and 12 h time periods, with an increasing percentage of devices being hit by the malware. Such attack rates could correspond to automated fast-spreading ransomware, such as NotPetya, and could also correspond to more modern human-operated ransomware. For example, in [40] an attack scenario is described in which an attack using Ryuk takes about five hours to complete, with most of the spread happening in the final hour using RDP from the domain controller. Other attacks use mechanisms like empire powershell, which may be detected by AV systems. Hence, within our attack model, we include both the speed of attack and the chance of success, which will relate to device configurations such as allowing RDP or AV signatures.

Given our intention to comparatively analyse the recovery techniques described in a meaningful way for real organizations, we now focus on a set of verification and validation methods for ensuring that the model is representative of the scenarios illustrated. We briefly summarize our verification and validation steps procedures.

When considering verification, we employed a mix of anti-bugging—if-clauses and counters that ensure the program workflow is not invalidating primary constraints, such as minimum and maximum possible speed on the network—model prototyping and simplified and structured walk-through iterations to continuously check that the model results match expected knowledge and expectations about the system, deterministic and non-deterministic variable assignation to flag possible unexpected behaviour, and event-tracing in the form of both time- and location-based logging.

Regarding validation, we based our model on two primary approaches: continuous validation via expert knowledge at every stage of the design development cycle and the use of real-world data manually gathered—additional information about the timings used can be found in Appendix A.

5.3 Results

Factors within the model such as when recovery is triggered and the movement of devices between locations are sampled from stochastic variables representing the environment. As such, each variant of the model (recovery scenario, attack

scenario, and success of the attack) is run 100 times and the graphs show the average performance, along with error bars showing the variability between the runs. The combination of these parameter ranges gives 144 different parameter configurations, each of which is run 100 times, resulting in a total of 14400 executions of the model. To obtain the results more quickly, we ran these across a number of computer cluster nodes. Most jobs completed successfully with a 3 GB memory limit; for the busiest scenarios, a 5 GB limit was sufficient.

Figure 4 shows how recovery speeds vary between the different methods under the different attack scenarios. We should expect that the network to saturate as mass recovery happens and hence slow recovery down. However, the model is showing that the recovery times are quite stable across the slower attack scenarios and recovery speeds only slow down with the fastest attacks. Even in this case, the recovery times increase gradually and, even with a severe attack, remain at a manageable speed.

The effectiveness of the USB-based recovery strategy is perhaps somewhat surprising, but the model assumes very good co-ordination within an office, with good availability of USB sticks, as well as good knowledge of how to use the process (which is less automated than HP 'Sure Recover'). These factors will depend on the preparedness and skill levels of the staff and hence would represent further enterprise-driven model parameters. The other factor with USB recovery comes from the availability of suitable equipment—Fig. 7 shows recovery failure rates between 5 and 40 systems, depending on the success of the attack (rather than speed).

As would be expected, HP 'Sure Recover Embedded' recovery is the quickest and most reliable mechanism. Having the embedded recovery within the mobile devices not only makes recovery of these devices quick it also significantly reduces the network load hence reducing the overall recovery times for the desktops dependent on the network even under the fast attacks. Even at a mix of 30% Mobile devices supporting HP 'Sure Recover Embedded' recovery the overall network load for mass recovery is significantly reduced. Suggesting considerable overall value for embedded recovery beyond just regular travellers.

The appendix contains a number of graphs showing how network contention affects different parts of our enterprise scenarios. Figures 8, 9, 10, 11, 12, 13 show considerable contention and slowdown with fast attacks on the office LAN. One question we wanted to show we can address using the model is the location of the recovery images. Figure 6 shows the results of placing an image server on the LAN within the large office as compared to Fig. 5 where all downloads go through the office internet connection. Given the speed of the LAN this leads to a massive improvement for the LAN connected desktops (the bottom line). The laptops recovery is slowed by the WiFi bandwidth limitations. Having the local image-server reduces recovery times significantly below those of the optimistic USB model, suggesting that a relatively cheap investment can significantly enhance recovery speeds (and without the security issues relating to USB based recovery).

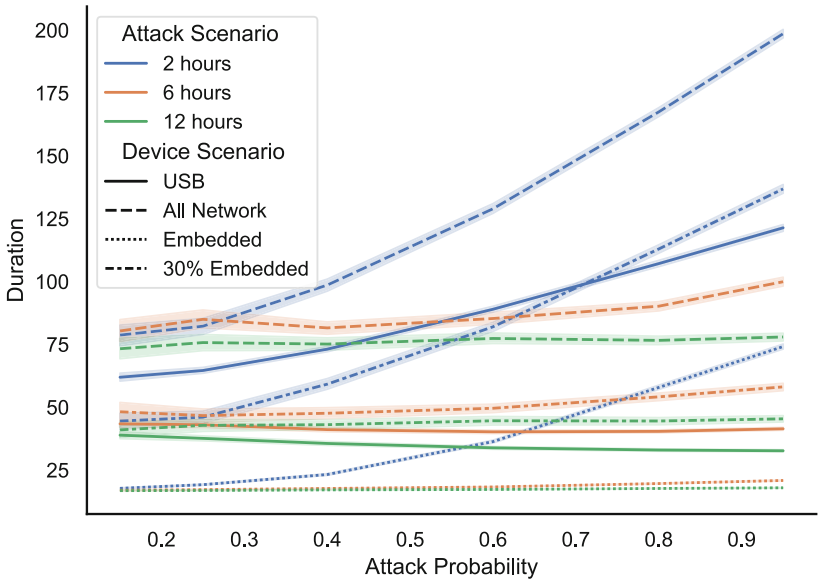

Fig. 4. All scenarios

5.4 Extending the Model

Within our the organizational model, we have a simple set-up in terms of recovery, in which we model recovery being triggered and the process happening, but do not consider the extent of the supporting management processes. These can be quite different for different enterprises. For example, in managing the USB-based recovery, some companies will have locked down the BIOS and will need administrators to be available to initiate recovery whereas others will allow users to perform these tasks. In addition, the skills and experience of users in different companies varies, as does the means to provide support, whether via help-desks or online.

The local reasoning concepts discussed in Sect. 4 talk about the ability to add resources and actions to the internal structure of a composed model whilst retaining the same interfaces. We can, for example, add additional actions and resource constraints to help handle USB recovery and have a smaller number of administrators who initiate USB recovery, and model the resource contention. We could have additional actions in which some users call help-desks and have additional resources to model help-desk operatives. Ideally, the latter would be based on either help-desk data or user studies to capture the resourcing required to support different likelihoods, under different modelled circumstances, that users will need help, given the desired times needed to help them.

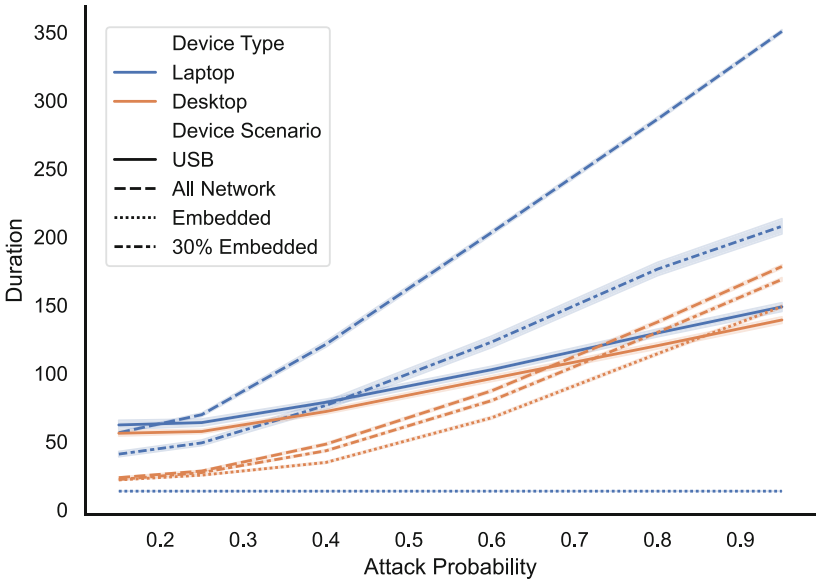

Fig. 5. Laptop vs Desktop; mean duration of recovery with remote image server; 2 h attack.

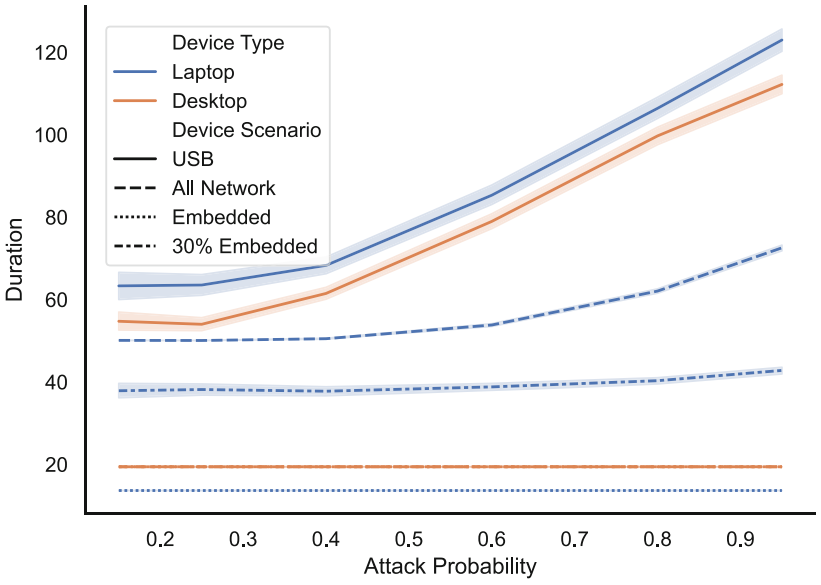

Fig. 6. Laptop vs Desktop; mean duration of recovery with local image server; 2 h attack.

6 Discussion

The time to recover—and whether recovery is even possible in a given location—are key measures for the success of any recovery system. However, from the enterprise's perspective the real concern is the cost of the impact and how a recovery solution can help mitigate those costs. It may be, for example, that costs of disruption to staff who are travelling are higher, as they need working systems to do sales presentations; or certain staff and offices may be doing time-critical tasks such as scheduling shipments. Such impacts can be very company-specific, but the model allows endpoint devices to be grouped with their locations and movements and with the need to recover to be defined. Detailed traces show the recorded recovery times. Thus, using the model, an enterprise can add value-information onto the overall recovery time information in order to compare the roll out of different strategies on disruption costs.

The model itself captures what happens when recovery is triggered over multiple systems within the enterprise. But there are other factors that must be taken into account for a full assessment of a recovery system to be deployed. For example:

- *Support During Recovery*—Critical aspects of recovery are the usability of any mechanism, how users know about mechanisms, and how easy they are to use. In Sect. 5.4, we discussed how extending the model with local reasoning can be used to model these further aspects. Here, evidence from usability studies will help in validating the model.
- *Security*—The security of a recovery process is critical. Security includes validating that recovery image deployed is that intended by the enterprise. Tools like HP 'Sure Recover' have signature checks and image locations built into the system such that the enterprise image is guaranteed to be deployed. When using mechanisms such as USB sticks, there can be an important security-versus-availability trade-off, in terms of locking down the boot process versus allowing users to recover when necessary.
- *Support costs*—As recovery solutions are rolled out, the costs for each solution must be assessed. These costs may be enterprise-specific; for example, support and images may be managed in various offices. We would argue that costs associated with USB based recovery can be higher—for example, in our model we assume a certain amount of IT support and preparedness within a large office. This contrasts with tools such as HP 'Sure Recover', which is more automated, so requiring configuration, but less active support.
- *Business Continuity Planning*—Part of the process of planning enterprise-scale recovery involves considering which threats are most concerning together with the importance of factors such as the time to recover. This reflects on the value discussion about how a company may be happy to take time to recover after a large scale outage. However, planning can help ensure certain key staff and tasks get priority, thus keeping essential operations running. This can, for example, be accomplished by having embedded recovery built into the laptops of travelling staff along with careful management of the placement of images in critical facilities.

These factors, and others, must be considered by system managers when making decisions about the recovery strategies and technologies to deploy. One way this might be achieved is by using a multi-attribute utility approach, where the preferences of system managers for various attributes—e.g. cost, time to recovery, information loss, and other measures of system performance—are used to determine a utility from the behaviour of the model (cf. [45]). The various policy and strategy choices can then be explored using the model, to determine which results in the greatest utility.

In this paper, we have modelled the re-imaging aspect of recovery, considering a case aimed at an enterprise managing its own image. In Sect. 2, we have outlined a wider space, including options for the enterprise to use modern management techniques and standard images. Such processes can be compared within our modelling framework; for example, varying image download times and changing the Windows install times to reflect differences in how users configure the system. Getting a user back and running also involves recovering applications and data. This can lead to additional network load as users download applications and data. However, cloud-synced storage devices, such as One Drive, will often configure file indexes and then download data as needed. The model can easily be extended to represent such additional network loads, but such loads will be very solution-dependent, and setting up the model requires careful planning.

7 Conclusion

Organizations are increasingly hit by malware, including rapidly spreading malware, which necessitates recovery strategies to eradicate the problems. Yet there are few tools that help enterprise decision-makers to understand recovery technologies and how they will react under such attack scenarios.

We have demonstrated how a modelling approach can help exercise critical aspects of the recovery problem, such as understanding network contention under different rapidly spreading malware attacks. Such tools can—for example, as demonstrated in the results—allow an enterprise to make choices about the placement of recovery images and about the value of having devices with embedded recovery support, both in recovering anywhere and reducing network load. This becomes possible because the modelling techniques allow models of different aspects of the IT stack to be composed. The local reasoning this enables will allow customization to match enterprise scenarios.

Further work will explore these issues—especially compositionality and local reasoning—theoretically, pragmatically, and empirically.

A Recovery Timings

In this section, we report some recovery times for the various methods used in our experimental section. These were generated from a few recovery runs within a single home environment (given the Covid 19-related restrictions prevalent in the UK at the time). These runs both helped inform parameters within the model and are presented to give the reader a feel for the different recovery techniques.

Table 1. 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.

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 2. 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.

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

B Additional Graphs

B.1 Recovery performance

USB-based recovery requires that users have a clean system to download an image, along with an available USB stick. Hence, within the model, we make assumptions around the levels of availability during recovery. Figure 7 shows the rate at which recovery is not possible under these assumptions with the different attack scenarios.

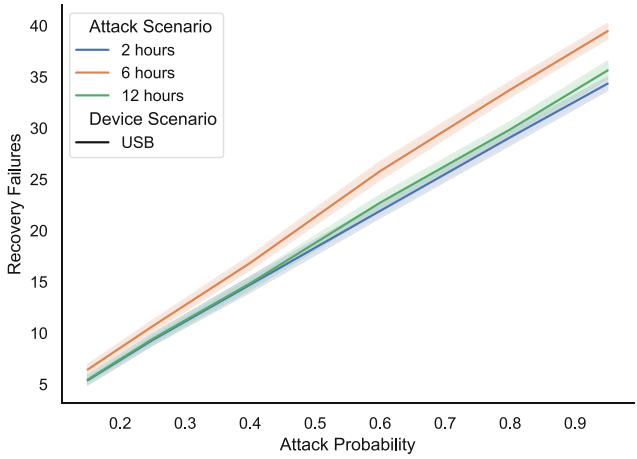

Fig. 7. USB failures

B.2 Network performance

Network contention is a large factor in recovery solutions, particularly when downloading images from the internet. Figs. 8 through 13 show the effective slow-down of the network because of contention. It is clear that where attacks happen quickly and with the attack happening within a two-hour period, the network load caused by recovery is significant. As attacks slow and become more gradual—say, as malware is distributed through email—then the network contention is reduced as the recovery load is spread over more time.

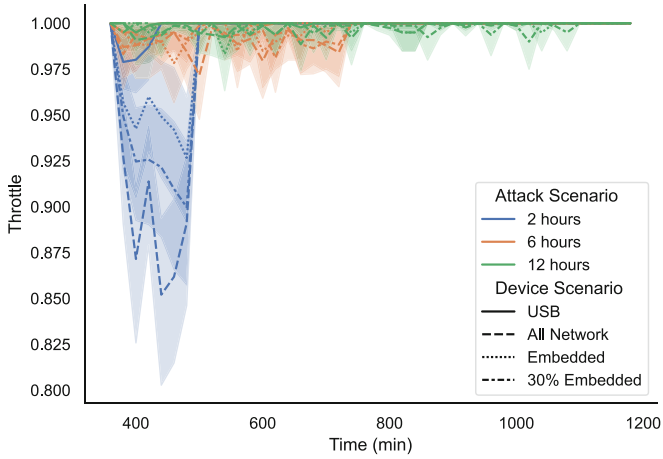

Fig. 8. Large office LAN throttling for 15% attack probability.

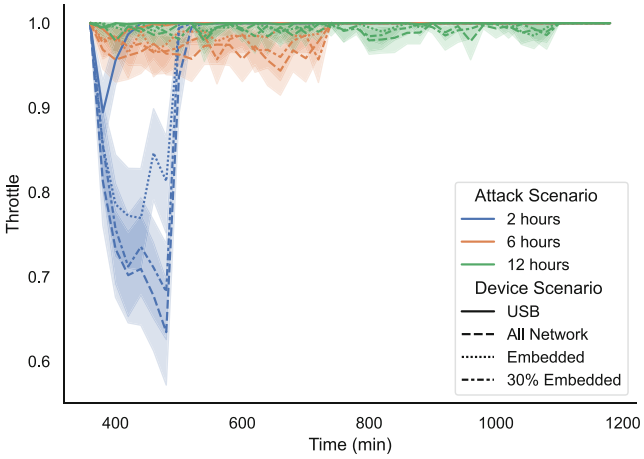

Fig. 9. Large office LAN throttling for 25% attack probability.

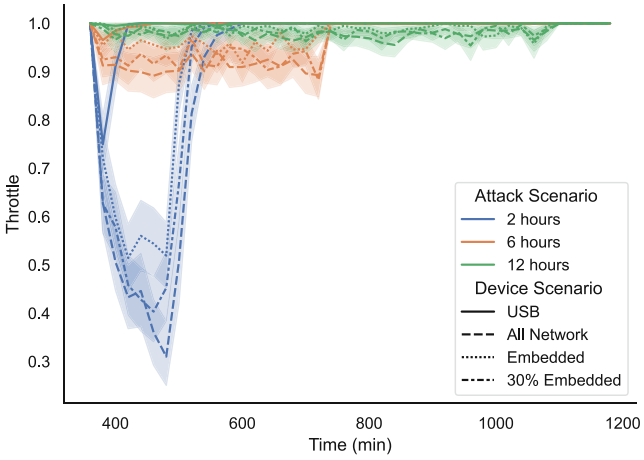

Fig. 10. Large office LAN throttling for 40% attack probability.

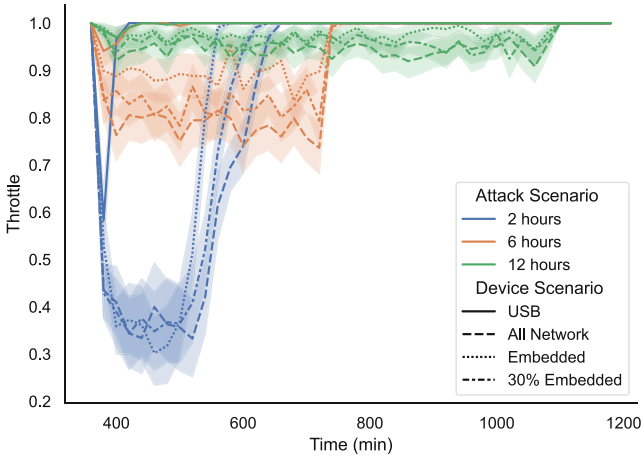

Fig. 11. Large office LAN throttling for 60% attack probability.

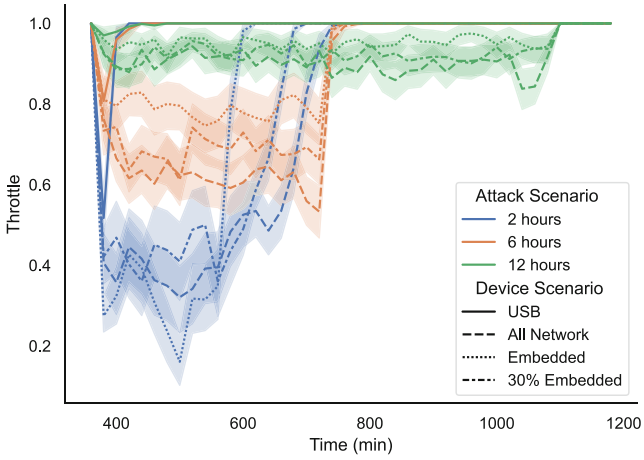

Fig. 12. Large office LAN throttling for 80% attack probability.

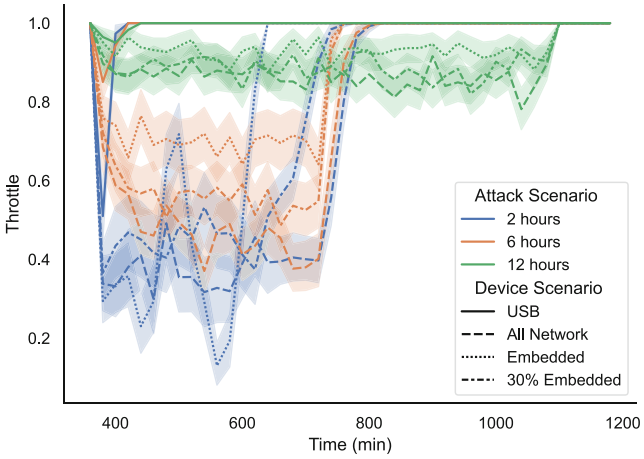

Fig. 13. Large office LAN throttling for 95% attack probability.

References

1. Anderson, G., Pym, D.: A calculus and logic of bunched resources and processes. Theoret. Comput. Sci. **614**, 63–96 (2016)
2. 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. (ed.) Economics of Information Security and Privacy III, pp. 213–238. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-1981-5_10
3. Beres, Y., Griffin, J., Shiu, S., Heitman, M., Markle, D., Ventura, P.: Analysing the performance of security solutions to reduce vulnerability exposure window. In: 2008 Annual Computer Security Applications Conference (ACSAC), pp. 33–42 (2008)
4. Beresnevichiene, Y., Pym, D., Shiu, S.: Decision support for systems security investment. In: 2010 IEEE/IFIP Network Operations and Management Symposium Workshops, pp. 118–125 (2010)
5. Binary Defense: Emotet Evolves With new Wi-Fi Spreader (2020). https://www.binarydefense.com/emotet-evolves-with-new-wi-fi-spreader/. Accessed 28 June 2021
6. Birtwistle, G.: Demos—Discrete Event Modelling on Simula. Macmillan (1979)
7. [blinded]: SysModels Julia Package. Accessed 10 May 2021
8. 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 June 2021
9. Caulfield, T., Pym, D.: Modelling and simulating systems security policy. In: Proceedings of SimuTools (2015)
10. cisa: Emotet Malware (2017). https://us-cert.cisa.gov/ncas/alerts/aa20-280a. Accessed 28 June 2021
11. Collinson, M., Monahan, B., Pym, D.: A Discipline of Math. Systems Modelling. College Publns (2012)
12. Collinson, M., Pym, D.: Algebra and logic for resource-based systems modelling. Math. Struct. Comput. Sci. **19**, 959–1027 (2009)

13. Dietz, G., Gillespie, N.: Recovery of trust: case studies of organisational failures and trust repair, vol. 5. Institute of Business Ethics London (2012)
14. Parizo, E.: Maersk CISO Says NotPeyta Devastated Several Unnamed US firms (2019). https://www.darkreading.com/threat-intelligence/maersk-ciso-says-notpeyta-devastated-several-unnamed-us-firms/a/d-id/1336558?page_number=2. Accessed 28 June 2021
15. Frazzon, E.M., Silva, L.S., Hurtado, P.A.: Synchronizing and improving supply chains through the application of cyber-physical systems. IFAC-PapersOnLine **48**(3), 2059–2064 (2015)
16. Gelenbe, E., Wang, Y.: Modelling the impact of cyber-attacks on web based sales (2019, Submitted for Publication)
17. Gibbs, M.: Your top it hates. Netw. World Canada **24**(8), N_A (2008)
18. Greenburg, A.: The Untold Story of NotPetya, the Most Devastating Cyberattack in History (2018). https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/. Accessed 28 June 2021
19. Hennessy, M., Milner, R.: On observing nondeterminism and concurrency. In: de Bakker, J., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 299–309. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10003-2_79
20. Hewlett-Packard Laboratories: Security Analytics. https://www.hpl.hp.com/news/2011/oct-dec/security_analytics.html. Accessed 10 May 2021
21. HP: HP SureRecover. HP (2021). https://www8.hp.com/h20195/v2/GetPDF.aspx/4AA7-4556ENW.pdf. Accessed 28 June 2021
22. Ishtiaq, S., O'Hearn, P.: BI as an assertion language for mutable data structures. In: Proceedings of POPL (2001)
23. Jung, J.Y., Blau, G., Pekny, J.F., Reklaitis, G.V., Eversdyk, D.: A simulation based optimization approach to supply chain management under demand uncertainty. Comput. Chem. Eng. **28**(10), 2087–2106 (2004)
24. Kral, P.: The Incident Handlers Handbook. SANS (2012). https://www.sans.org/reading-room/whitepapers/incident/incident-handlers-handbook-33901. Accessed 28 June 2021
25. LogRhythm Labs: A Technical Analysis of WannaCry Ransomware (2017). https://logrhythm.com/blog/a-technical-analysis-of-wannacry-ransomware/. Accessed 28 June 2021
26. Lynch, N.A., Tuttle, M.R.: An introduction to input/output automata. CWI Quart. **2**, 219–246 (1989)
27. 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 June 2021
28. 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 June 2021
29. Microsoft: Microsoft Digital Defense Report - Sept 2020. Microsoft (2020), https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWxPuf/. Accessed 08 Aug 2021
30. 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 June 2021

31. 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 June 2021
32. Microsoft: Overview of Windows Autopilot. Microsoft (2021). https://docs.microsoft.com/en-us/mem/autopilot/windows-autopilot. Accessed 28 June 2021
33. Milner, R.: Calculi for synchrony and asynchrony. Theor. Comput. Sci. **25**(3), 267–310 (1983)
34. Milner, R.: Communication and Concurrency. Prentice Hall, New York (1989)
35. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press, Cambridge (2009)
36. Milner, R.: Bigraphs as a model for mobile interaction. In: Corradini, A., Ehrig, H., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2002. LNCS, vol. 2505, pp. 8–13. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45832-8_3
37. O'Hearn, P.: Resources, concurrency, and local reasoning. Theor. Comput. Sci. **375**(1–3), 271–307 (2007)
38. O'Hearn, P., Reynolds, J., Yang, H.: Local reasoning about programs that alter data structures. In: Fribourg, L. (ed.) CSL 2001. LNCS, vol. 2142, pp. 1–19. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44802-0_1
39. Pym, D.: Resource semantics: logic as a modelling technology. ACM SIGLOG News **6**(2), 5–41 (2019). https://doi.org/10.1145/3326938.3326940
40. Report, T.D.: Ryuk in 5 Hours. The DFIR Report (2020). https://thedfirreport.com/2020/10/18/ryuk-in-5-hours/. Accessed 08 Aug 2021
41. Reynolds, J.: Separation logic: a logic for shared mutable data structures. In: Proceedings of LICS (2002)
42. Rhodes, C., Bettany, A.: Windows Installation and Update Troubleshooting. Apress, New York (2016)
43. Hurley, S., Sood, K.: NotPetya Technical Analysis Part II: Further Findings and Potential for MBR Recovery (2017). https://www.crowdstrike.com/blog/petrwrap-technical-analysis-part-2-further-findings-and-potential-for-mbr-recovery/. Accessed 28 June 2021
44. Stirling, C.: Modal and Temporal Properties of Processes. Springer, Heidelberg (2001). https://doi.org/10.1007/978-1-4757-3550-5
45. Caulfield, T., Pym, D.: Improving security policy decisions with models. IEEE Secur. Priv. **13**(5), 34–41 (2015)
46. M.D.T.I. Team: Human-operated ransomware attacks: a preventable disaster. Microsoft (2021). https://www.microsoft.com/security/blog/2020/03/05/human-operated-ransomware-attacks-a-preventable-disaster/. Accessed 08 Aug 2021
47. Tweneboah-Kodua, S., Atsu, F., Buchanan, W.: Impact of cyberattacks on stock performance: a comparative study. Inf. Comput. Secur. (2018)
48. 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 08 Aug 2021
49. Yang, H., O'Hearn, P.: A semantic basis for local reasoning. In: Nielsen, M., Engberg, U. (eds.) FoSSaCS 2002. LNCS, vol. 2303, pp. 402–416. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45931-6_28