Array 26 (2025) 100391

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

STATOS: A portable tool for secure malware analysis and sample acquisition in low resource environments

Alexander Cameron [a], Abu Alam [a], Nasreen Anjum [b], Javed Ali Khan [c], Alexios Mylonas [c,*]

[a] School of Computing & Engineering, University of Gloucestershire, The Park, Cheltenham, GL50 2RH, Gloucestershire, UK
[b] School of Computing, University of Portsmouth, University House, Portsmouth, PO1 2UP, Hampshire, UK
[c] Cybersecurity and Computing Research Group, Department of Computer Science, University of Hertfordshire, College Lane Campus, Hatfield, AL10 9AB, Hertfordshire, UK

ARTICLE INFO

Keywords:
Cyber attacks
Malware analysis
Static analysis
Cyber security

ABSTRACT

Malware poses a significant security threat to organisations worldwide, particularly in environments with limited resources. Static analysis has emerged as a crucial technique for gaining insights into malware, but it often requires specialised hardware and software, which can be a barrier for organisations facing financial or supply constraints. To address these challenges, this study presents a Static-Analysis Operating System (StatOS), a portable Linux derivative operating system designed for static malware analysis. StatOS can be executed from a USB device, allowing organisations to perform efficient, user-friendly, and secure malware analysis even on underpowered hardware. This study contributes a practical solution to field analysis of malware within low-resource environments, providing a model and requirement data for future developments in portable cybersecurity tools. The tool was validated through a combination of expert feedback using the Delphi method and security assessments, including Monte-Carlo simulations and Common Vulnerabilities and Exposures (CVE) evaluations. Results indicate that StatOS meets and exceeds key performance requirements, with 100% of surveyed cyber specialists agreeing on its effectiveness, and 80% indicating they would use StatOS in forensic investigations.

1. Introduction

Throughout the history of computer systems, constant competition has existed between criminals attempting to exploit computer systems for their advantages and organisations vying to prevent such attacks. As technological advances have accelerated, the suite of tools available to cybercriminals has expanded. Cybercriminals no longer need to manually infect and perform actions on compromised systems; instead, they can integrate these tasks into malicious software, known as malware [1].

Malware can be packaged and delivered in various formats to deceive individuals about the software's true purpose. Once executed, the cybercriminal or the malware itself may perform actions such as encrypting files, destroying data, or covertly collecting information on a target [2,3]. While reactive protections, such as endpoint protection software capable of remotely removing suspicious files and antivirus removal procedures are available, proactive protections that can be utilised before malware execution are crucial. These proactive measures can potentially eliminate an infection before it fully occurs, thereby reducing the potential damage to an organisation from a cyber attack [4].

It is essential that organisations have the technical capabilities to analyse malicious software to continually improve security and prevent malware infections that could cause financial and reputational damage [5,6]. By performing such analyses, organisations can determine with higher reliability if the software is malicious or disreputable. Additionally, they may discover indicators within the suspicious software that could be used to identify similar malicious software across the organisation's network, helping to prevent future infections [7].

However, performing such analyses often requires specialist equipment, including forensic operating systems, disassembly tools, and dedicated, computationally capable hardware, such as systems with write-protected storage devices to prevent infections [8,9]. In the event of a cyber attack requiring forensic investigation, individuals must act quickly to preserve and acquire digital evidence [10–12]. Rapid sample

* Corresponding author.
E-mail address: a.mylonas@herts.ac.uk (A. Mylonas).

https://doi.org/10.1016/j.array.2025.100391
Received 10 November 2024; Received in revised form 28 February 2025; Accepted 22 March 2025
Available online 1 April 2025

acquisition and investigation are recognised as significantly enhancing an organisation's ability to recover from such attacks [13,14].

1.1. Problem statement

Despite the critical importance of rapid and effective malware analysis, organisations often do not proactively invest in the necessary hardware and software solutions to achieve this goal or are unable to invest, e.g., due to lack of resources. This lack of investment results in a diminished capacity to perform thorough investigations during an attack, which in turn weakens organisational security [15,16]. Further complicating matters, digital forensics often requires an abundance of software tools and the necessary resources to accommodate these tools, presenting challenges for organisations with limited capacity [17].

Static analysis, a procedure in which items suspected of harbouring malware are dismantled from their binary presentation and investigated for potential indicators, offers a less resource-intensive method than dynamic analysis. Static analysis can identify threats without executing the malware, thereby reducing the risk to the organisation [18–21]. However, due to financial and supply constraints, organisations may struggle to dedicate systems solely for static analysis. Additionally, digital forensics experts often need to switch between multiple systems, each running different operating systems to utilise various tools depending on the task [22].

Reusing existing computer equipment for sample acquisition and analysis instead of investing in new hardware suggests a gap in the cybersecurity field for a reusable and potentially portable operating system. Such a system could be deployed on existing hardware to perform static analysis without requiring additional capital expenditure [23,24]. Based upon this rationale, the study aims to develop and validate a portable Static Analysis Operating System (StatOS), designed to alleviate the hardware and resource constraints organisations face during static malware analysis. The developed tool and research strategy defined will assist in answering the research question ***Can a portable malware analysis tool enhance the ability of organisations to respond to malware attacks?***

The paper firstly seeks to understand how StatOS can be designed and developed to be a portable, low-resource tool that effectively performs static malware analysis on underpowered hardware while maintaining high performance and security, answered in Section 5. Secondly, the researchers sought the opinions of digital forensic experts on what essential features and static analysis tools should be integrated into StatOS to ensure usability and effectiveness, answered in Sections 3.1 and 4.3. Thirdly, the effectiveness, robustness, and potential vulnerabilities of StatOS were validated through expert feedback and security assessments to ensure its reliability in real-world applications, visible within Sections 5.2 and 7.

1.2. Contribution and organisation

This paper makes the following contributions:

1. **Development of StatOS:** The creation of StatOS as a portable, low-resource tool for malware analysis represents a practical solution to the hardware limitations commonly faced by organisations, and provides a model for the development of similar tools in the future.
2. **Validation of StatOS:** Through a comprehensive validation process involving the Delphi method and expert feedback, the study demonstrates the tool's effectiveness, performance, and usability, providing evidence that StatOS can meet industry needs.
3. **User-Centred Design:** By incorporating feedback from digital forensics experts throughout the development process, the study ensures that StatOS is aligned with user expectations and practical requirements, enhancing its applicability in real-world scenarios.

This paper is organised as follows: Section 2 provides a critical evaluation and comparison of three state-of-the-art static analysis tools – GHIDRA, IDA, and BinaryNinja – highlighting their features, strengths, and limitations within the cybersecurity ecosystem. Section 3 discusses the strategy employed to understand the requirements and perform an evaluation of the tool. Section 4 outlines the formation of requirements from research and industry expert opinions. Section 5 translates these requirements into a functional tool. Section 5.2 presents the integrity testing of the tool through Monte-Carlo simulation and analysis of active CVEs. Section 7 employs Delphi-method semi-structured interviews with industry experts to compare the tool against its requirements. Finally, Section 8 summarises the findings of the research and discusses the potential impact of StatOS on the field of malware analysis and cybersecurity.

2. Evaluation and comparison of state-of-the-art static analysis tools

Specialist software tools are available to assist in the process of static analysis, such as BinaryNinja, GHIDRA, and IDA. However, tools such as these that perform static analysis are fragmented across the cyber security ecosystem. This section provides a critical evaluation and comparison of three state-of-the-art static analysis tools—GHIDRA, IDA, and BinaryNinja—highlighting their features, strengths, and limitations within the cybersecurity ecosystem. Table 1 provides a consolidation of the critical evaluation of these tools, comparing the limitations and features essential for static analysis.

2.1. GHIDRA

GHIDRA is an open source, reverse engineering framework, written in Java by the National Security Agency of America and recently declassified to the public [25]. As suggested by [26] while other reverse engineering tools, such as IDA exist with enhanced capabilities, GHIDRA elevated its position as a static analysis tool due to its open source extensible framework. In addition to GHIDRA being open source, GHIDRA is also free for private and commercial use, allowing organisations to make use of the software without a capital cost as well as permitting organisations not to publicise their own customised modifications they make to the software [27]. Nevertheless, GHIDRA has limited capabilities compared to tools such as IDA, which have been commercially used for many years, which has led to greater uptake and support when compared to GHIDRA, which is comparatively new.

GHIDRA is capable of simultaneously loading multiple binary files into a single project for investigation, allowing multi-file investigations. For static analysis, this feature is exceptionally useful in situations where one piece of malware may deposit additional files such as libraries or further payloads. These additional files could be analysed in conjunction with the original file to determine the relationship between the different components and potentially the actions taken [28]. GHIDRA also supports custom add-ons which can be created by users to enhance its abilities and allows multiple users to share projects, potentially containing multiple files which can be worked on by both users.

2.2. IDA

IDA (Interactive Disassembler) has existed since 1995 and has matured considerably, with support for a vast number of architectures and files [29]. IDA is also available with support for customers who have purchased the product, which can assist individuals in performing static analysis and debugging certain issues [30].

To improve the process of static analysis, IDA also supports variable mapping in both its compiler and interface. This allows users to create user-friendly names of common values that occur throughout a suspicious file to begin to piece together an understanding of the file

Table 1
High level feature comparison of GHIDRA, IDA and BinaryNinja tools used for static analysis.

Comparisons	GHIDRA	IDA	BinaryNinja
Summary	Java based reverse analysis framework created by the NSA.	C++ reverse analysis environment created by Hex-Rays	C+,C, Python static analysis tool created by BinaryNinja
Supports x86, x86 64?	Yes	Yes	Yes
Open Source?	Yes	No	No
Approximate Tool Maturity	Very New (First Released 2019)	Very Mature (First Released 1995)	New (First Released 2016)
Specialist Features	Effective on files over 1GB without substantial performance penalty. Multiple binaries scan be loaded at once.Multi-user capabilities.	Huge range of supported languages, architecture and file types. Customer support and variable mapping.	Cloud capabilities, programmatic API for automation of analysis.

they are investigating. Such a feature is not available by default within GHIDRA and instead users must rely on the open source community developing additions to service this requirement or develop their own additions [31].

However, while the software is far more mature than that of GHIDRA and would be desirable for inclusion, the lack of open source availability and licensing constraints limit the effectiveness of IDA as a potential solution. Each organisational instance of IDA requires an active licence, which may be impractical for static analysis in incident response scenarios.

2.3. BinaryNinja

Binary Ninja is a recent reverse engineering toolset, equipped with cloud based analysis mechanisms to enable users to offload potentially computationally intensive tasks from their own hardware. While both GHIDRA and IDA allow decompilation, the ability to work on a project with other users potentially across a wide geographical location is an invaluable feature for organisations with a geographically dispersed workforce. BinaryNinja allows reverse engineering to take place in a cloud environment. The software is free to use, provided an account is created and allows users to collaborate with others on static analysis projects, while also allowing statistical graphs to be dynamically generated from interrogated files [32].

However, any binaries being investigated must be uploaded to BinaryNinja, which may be difficult for organisations who are bound by confidentiality. Additionally, due to the security arrangements commonly in place within malware analysis environments, it is likely that air gapped networks and computer systems would be used for malware analysis to prevent malware reporting out to its authors that it has been detected [33]. This would negate any benefit of cloud services, as while their features would be useful, air gaps or other connection limitations may prevent connectivity and usage of the service. Therefore, with no central Linux distribution for this purpose exclusively, combined with current shortages of computer hardware, organisations seeking to perform static analysis must re-use multiple existing computer systems to support the multiple operating systems and tools required.

2.4. Operating Systems

To support the usage of such tools and acquisitions, a range of Linux operating systems exist. Such operating systems encompass commonplace cyber security tools and aim to provide a holistic development and testing environment to cyber security experts. Examples of such operating systems include Kali Linux, a Linux distribution focused around cyber security and penetration testing [34]. Kali Linux contains over 20 tools aimed at static analysis, however not all of these tools are immediately available and may require internet connectivity to download the tools from Kali Linux's repositories. Additional Linux derivatives such as the SIFT workstation created by the SANS Institute are directly aimed at digital forensics experts, containing hundreds of small to medium sized software packages to aid forensic analysis [35].

The tool being developed, StatOS, aims to facilitate lightweight malware analysis through the adoption of a minimal operating system that can be deployed to infected computer systems to perform both acquisition and analysis in the field by responders. In this way, a compromise can be reached between resourcing requirements, tooling and functionality to design a tool capable of both response and analysis. Table 2 summarises the operating systems and their aspects.

3. Methodology

To gain a thorough understanding of which security tools should be added to the solution, primary research of existing static analysis software favoured by digital forensics experts will be undertaken. The findings will inform the evaluation process of the product through Strengths, Weaknesses, Opportunities and Threats (SWOT) analysis to accurately understand potential areas of improvement for future development. Results of the primary research will be compared against existing sources to determine if the experts surveyed in the primary research agree with the general sentiments expressed by existing sources and if not, what specific differences are visible between pre-existing research and the research undertaken within the project. A full flowchart of the employed methodology can be seen in Fig. 1. The relevant data protection legislation within the United Kingdom and the ethical operating procedures of the researchers institutions were followed when performing research using surveys and interviews.

3.1. Sourcing requirements

To design StatOS, it was necessary to understand the requirements of a wide range of cyber security and digital forensic experts to justify design decisions such as the type of operating system, graphical environments and tool suites to include. To collect the aforementioned information, a digital survey was created and circulated between individuals working in computer science, cyber security and digital forensics, with snowball sampling assisting in collating further respondents.

3.2. Result processing

After collecting the requisite information, the results were obtained in a comma separated values (CSV) format, which was subsequently analysed using a python script. The script was capable of interfacing with a given CSV results file and scoring respondents based on their malware forensics experience and job title in terms of relevance to malware forensics. The resultant technicality was calculated by summing the scores of the above answers, with respondents being categorised into low, medium and high levels of technicality in regards to malware analysis.

Table 2
A table comparing high level features of commonplace operating systems used for static analysis and digital forensics.

Comparisons	Kali Linux	SIFT workstation
Overview	Kali Linux advertises itself as a Linux distribution for cyber security professionals, containing a wide range of software suites aimed at all aspects of cyber security. Static analysis and digital forensics make up a portion of the software suites used.	The SIFT workstation is a suite of tools and pre-packaged version of Ubuntu, specialising in digital forensics, file recovery and malware analysis.
Parent/Derivative	Debian Testing Branch	Ubuntu
Target Audience	Cyber Security Experts	Digital Forensics Experts
Forensic Analysis Tools Present	23	>50
Live CD Capability?	Yes	No
Source	Gunawam, Lim, Zulkurnain, and Kartiwi, 2018	Sans Institute, 2012

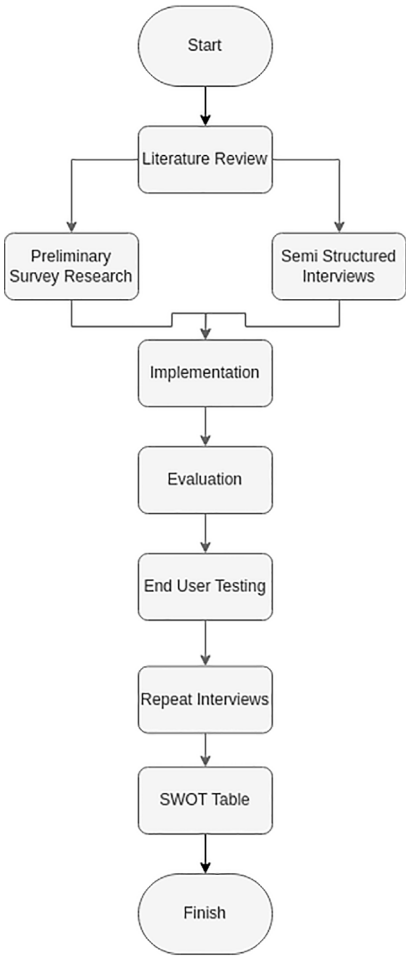

Fig. 1. A flowchart of the research methodology to be used. Following a literature review, surveys will utilise a combination of snowball sampling and convenient sampling of digital forensics experts to inform the implementation of StatOS. Once implemented, StatOS will be evaluated using the Delphi method with digital forensic experts to support in answering the research question.

3.3. Result categorisation

Participants were asked to rank the importance of several high-level requirements and features they would expect to be present in an operating system tool supporting malware analysis. To facilitate understanding, the authors have included these terms and the definitions used:

1. **Portability:** The ability to utilise the tool in different environments.
2. **Performance:** The speed of the tool to complete forensic taskings.
3. **User Frendliness:** The ease of use of the tool.
4. **Cost:** The expenditure required to deploy and use the tool.
5. **Integrity:** The reliability of the tool to maintain system stability.

1. **USB Persistence:** The ability of the tool to maintain data following use.
2. **RAM Capabilities:** The ability of the tool to deploy itself as a RAM-disk.
3. **Analysis Tools:** The collection of malware analysis tools present on the operating system.
4. **Programming Tools:** The collection of programming tools present on the operating system.
5. **Run on Low Hardware:** The ability of the tool to operate on low-end computer hardware.

3.4. Method justification

The approach of collecting requirements and insights from a software project target audience and incorporating this into software projects is a widely recognised approach that enables development teams to ensure that a project aligns with the expectations of potential users [36–40]. Advancements such as Agile software development and crowdsourcing requirement methodologies are also capable of integrating requirements through surveying users, leading to a greater alignment of developers and user bases when compared to not using such methods [41,42].

Furthermore, the use of the Delphi method to validate an engineering artefact meets requirements is a well established method to reach a consensus against a target group [43]. Whilst other consensus methods have seen widespread use such as the Nominal Group Technique, such methods are generally recognised to be more suitable to research attempting to explore ideas or propose multiple solutions to a problem [44]. As the researchers wish to answer several fixed questions, the Delphi method was chosen due to its widespread usage within the domain of cyber security and software engineering, in which it has been used comprehensively to validate the designs of both software and key frameworks [45–47].

Whilst both methods are effective at different stages, they can present disadvantages. Requirement sourcing typically requires a large sample size, which is difficult to acquire and also commonly leads to only surface-level requirements being discovered, as the level of detail and subjectivity is marginalised through a survey [48]. In contrast, the Delphi method typically requires a much smaller sample size as each of the interviewees are selected based on key relevance to the research, enabling a greater level of granularity and detail when compared to

Table 3
Participant demographics of digital forensic experts within the research.

Demographic data	Point survey	Interviews
Respondent Ages	18–55	18–26
Geographical Location	United Kingdom	United Kingdom
Ethnicity	Not Collected	Not Collected
Gender	Not Collected	Not Collected
Experience in Computing	Moderate to Substantial Experience	Substantial Experience
Experience in Cyber Security	Minor to Substantial	Substantial Experience
Experience in Malware	Minor to Substantial Experience	Moderate to Substantial Experience

requirement crowdsourcing. This detail can present analysis difficulties as it may not be directly quantifiable or generalisable, however may provide greater context in answering the research question within Section 1.1. By combining requirement crowdsourcing and interviews through methodologies such as the Delphi-method, a balance can ensure that the solution created aligns with user requirements at the design stage and allows for expert scrutiny of the solution [49].

To expand upon the outlined methodology, 50 experts will be selected to support requirement crowdsourcing, based upon a combination of convenient sampling and snowball sampling of digital forensics experts sourced from the UK [50]. These sampling methods have been chosen due to the range of experts available in the local area. Once selected and consent is given, experts will complete an online survey, outlining the capabilities of the product. Once the product has been implemented, experts will then, using a Likert scale, rank the effectiveness and robustness of the product. A Likert scale has been used to reflect and measure the granularity of answers given [51].

Additionally, the Delphi-method has been chosen to ensure a consensus is reached across a range of industry experts, reducing the impact of potential selection bias from the researcher. Interviewees were selected based upon the relevance of their role to the subject matter, accessibility to the researchers and expertise within fields of malware, rather than other factors, which ensures that the technical capabilities of the respondents align with the target audience of StatOS [52–54].

Information has been provided upon the demographics of participants of the research within Table 3.

To inform the final SWOT analysis, Monte-Carlo methodologies will be used with automated penetration testing software to provide insight into StatOS's robustness. Monte-Carlo methodologies have been selected as an appropriate method due to the ability to repeatedly test an item to understand its characteristics [55]. After as many rounds of penetration testing could be reasonably performed, the average number of vulnerabilities and their CVSS impact score will be analysed to give an overall impression of the solutions dependability [56].

After completion of the surveys and interviews, the Delphi method will be deployed to in conjunction with industry experts selected through convenient sampling and snowball sampling to evaluate the product. The Delphi method has been selected as it will allow experts to form a majority consensus, which will greatly assist in the creation of the final SWOT analysis [57].

4. Requirement sourcing results

The following section will discuss the results of sourcing requirements for StatOS from industry experts, followed by the processing of the aforementioned results to inform the implementation process. Opinions and sentiments regarding operating systems, user interfaces and favoured static analysis tools will be evaluated to form a table of key requirements.

4.1. Operating system findings

It has been found that low technicality groups place a higher value on programming tools than malware analysis toolsets. All respondents within the low technicality group had not performed any malware analysis. Therefore, it is likely that acquiescence bias could be impacting respondents; [58] discusses the tendency for likert scales such as the scales used within the survey to cause respondents to select seemingly positive answers, regardless of the justification behind such selections. Fig. 2 displays these results.

All technicality groups had placed a high value on performance, however it is notable that high technicality groups place a higher value on portability than all other groups, potentially due to the nature of digital forensics and computer incident response engagements often requiring physical relocation to a customer site, where portability would be useful [59]. Medium technicality groups place a lower level of value on user friendliness than all other groups, which is validated by the group's high uptake of terminal usage compared to both low and high technicality.

Low and high technicality groups are almost congruent, mirroring each other in terms of all aspects except portability. Uniquely, medium technicality respondents value the operating systems integrity at a lower level than low and high technicalities. This differs from other groups and both industry and academic consensus, as for malware analysis, isolation and integrity can be deemed highly essential due to the risk of infection [60].

To understand potential features to implement into StatOS, respondents were asked to rank key features and how essential they believe the features to be. From such questions, the following results were collated and are visible in Fig. 3.

As can be seen in Fig. 3, all technicality groups value the majority of features. However, medium technicality groups diverge away from ramdisk capabilities, while low technicality groups place the highest value on programming tools. Such differences could be due to medium technicality respondents having performed a minimal level of malware analysis, potentially being impacted by the ambiguity effect. When deprived of key information to support a decision, respondents may prefer options they are familiar with and understand, rather than options that, while potentially the same or better, are not well understood [61]. The higher value of programming tools among low technicality groups can likely be attributed to the fact the group has not performed malware analysis in any form. Without experience of malware analysis it is likely the respondents are falling-back to tools they understand the importance of [62].

Other notable differences between groups would be that high technicality respondents place a higher value on USB persistence than medium and low technicality groups. This could potentially be because experienced malware analysts may wish to save ongoing investigations and progress to the device, a process commonplace for more advanced or multi-stage analyses, which may not have been identified by other groups [63].

4.2. Interface findings

Regarding interfaces, participants of all technicality groups favour some form of GUI and terminal usage. The low technicality group represents the lowest share of participants who prefer terminals exclusively, while the high technicality group represents the highest share of participants preferring terminals. This is likely due to terminals inherent verbosity and lack of abstraction, ensuring advanced users receive more detailed information than they may receive if performing a similar action through a GUI interface [64]. Results illustrating the above have been included in Fig. 4:

It is notable that terminals never exceed $\frac{1}{4}$of responses, with the majority of respondents in both high and low technicalities preferring GUI's over terminals. Medium technicalities prefer a combination of

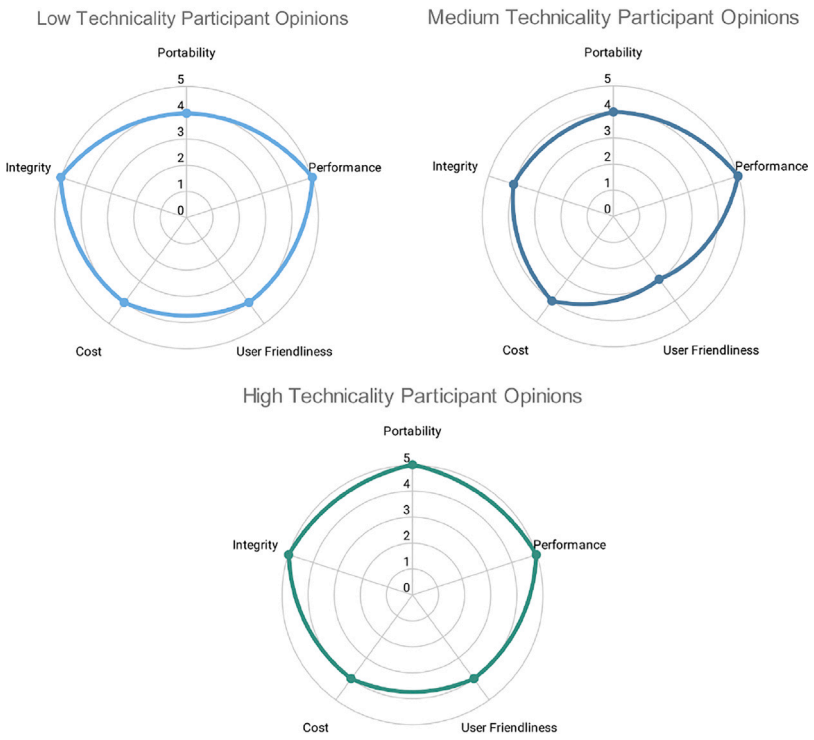

Fig. 2. Radar charts identifying each technicality group's opinions on key operating system aspects.

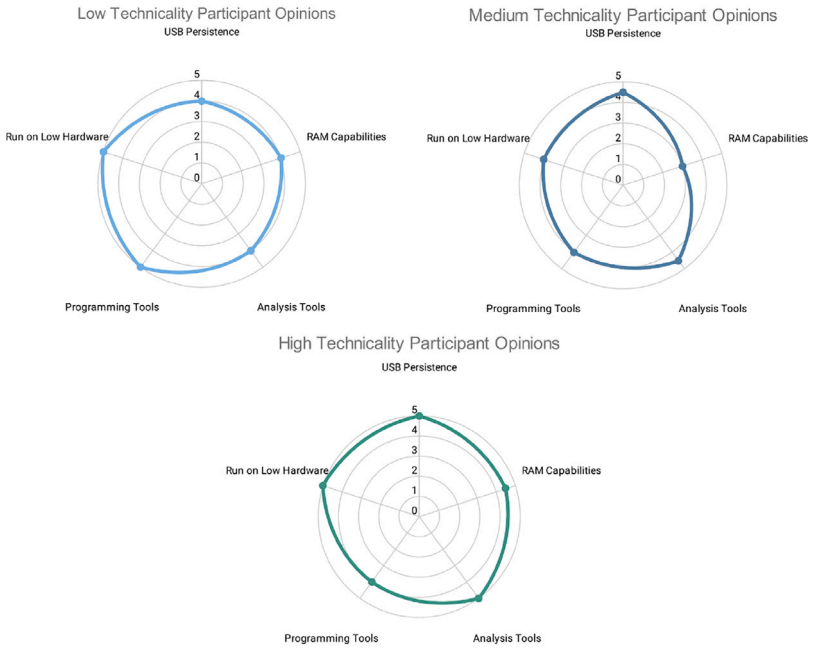

Fig. 3. Radar charts identifying each technicality group's opinions on key operating system features.

both GUI's and terminals, which is the second-most preferred option in all respondent groups. Based on these findings, StatOS must include some form of graphical user interface in addition to conventional terminal interfaces, as all demographics appear to benefit from the inclusion of such interfaces [65].

In respondents who identified GUI's as their primary interface of choice, the survey opened an additional questioning path in order to understand the specific types of GUI respondents preferred and why. From such questioning, the following results were collected in Fig. 5.

Specific GUI's such as XFCE and Gnome3 mentioned in medium and high technicality groups, making up the second-highest selection after Windows-like GUI's. Low technicality groups mention MacOS-like GUI's and KDE, however the frequency of respondents identifying these are comparatively low to the volume of low technicality respondents selecting windows-like GUI's. Generally, environments that are described as "windows-like" are preferred, presumably due to the respondents familiarity with the Microsoft Windows software suite and respondents adhering to this choice [66]. Respondents were asked in a free-text

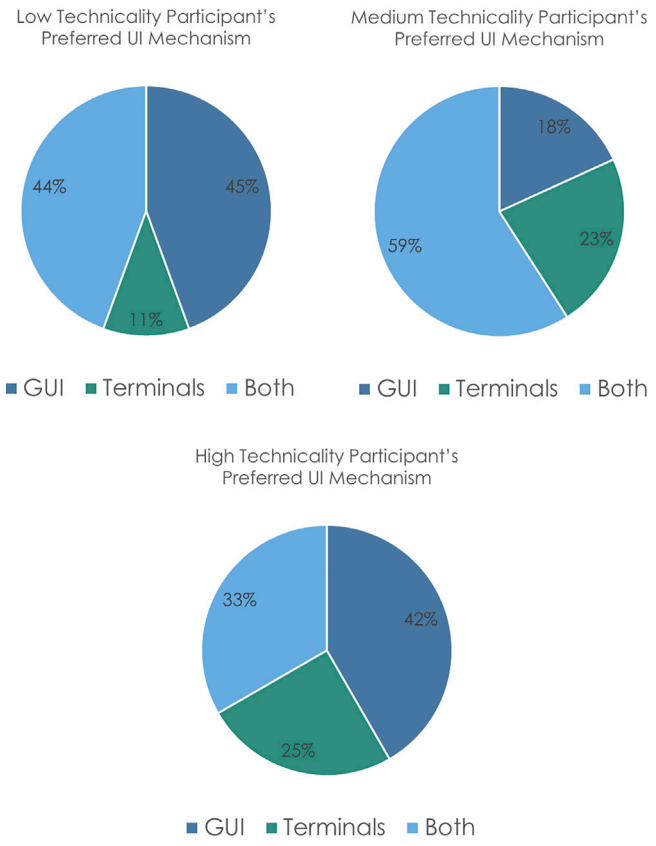

Fig. 4. Pie charts illustrating the breakdown of preferences to GUI's and terminals.

field to identify why they preferred GUI's, with the resulting answers collated and analysed through a word-frequency cloud which has been displayed in Fig. 6.

4.3. Tool findings

In relation to tools, GHIDRA and IDA were the top two tools in both the low and high technicality groups, with the medium group favouring BinaryNinja. Paradoxically, while this software suite is most favoured in medium technicalities, the software is least favoured in both low and high technicalities; a potential theory could be that the advanced interface options may be found to be difficult for lower technicalities due to an expectation of knowledge, whilst also obscuring more advanced options behind wizards and menu options. To draw parallels with this idea, [67] suggests that specialised CLI interfaces are often more performant and reliable, however have a long learning curve. Portions of this thinking may be applicable to advanced GUI's such as those of IDA and GHIDRA. The results have been displayed as a stacked bar graph in Fig. 7.

As can be seen, while the data supports the fact that BinaryNinja is most favoured in medium technicality groups, the data also supports the fact that BinaryNinja appears to be the least favoured of all the tools in the same technicality group; investigating the raw data resolves this anomaly, displaying that respondents opinions varied considerably for this data point. Users were prevented from selecting the same tool more than once in any column, thereby preventing this unusual data from being generated by error. Instead, it appears that the balance of opinion in the medium technicality group is very balanced in this regard, with some respondents in favour of BinaryNinja, while others are against this. Based upon the results discussed, the following core requirements of StatOS could be created, visible in Table 4.

Table 4
A table of requirements, based upon the research conducted.

Requirement No	Requirement
1	Support underpowered hardware
2	Include familiar GUI's and terminals
3	Inclusion of malware and programming tools
4	Portability and performance to be prioritised
5	System must provide appropriate integrity
6	System must be user-friendly to target audience
7	USB persistence and ramdisk capabilities

5. OS implementation

To support the discovered requirements, we utilised open source software from the GNU Project, the Linux Kernel and Linux From Scratch project, with other supporting open source software included [68,69]. The minimal amount of software packages required to create a Linux based operating system suitable for static analysis tasks was determined by analysing the minimum requirements of Linux operating systems. Once identified, relevant packages were configured, compiled and incorporated into the solution, with this process repeated until a functional tool was created.

Ensuring StatOS remains portable and highly performant led to the design decision to utilise a USB 3.0 compliant thumb drive. As the devices are physically compact and can be manufactured in a variety of sizes, the portability requirement of StatOS would be satisfied. Furthermore, USB 3.0 devices are backwards compatible with USB 2.0 compliant ports, found on the majority of computer systems in the past few decades. USB 3.0 devices support some of the highest available speeds for USB devices, ensuring performance is maintained [70].

The design process utilised a virtualized disk image, which was iteratively written to a physical USB 3.0 device during development and testing. The partitioning scheme relied on three partitions. A design decision was taken to create a separate partition to store "boot" files such as kernel images and bootloader modules to avoid potential erasure of the key kernel images used to boot the device. As the composite files are relatively small, the partition does not need to exceed 250 mebibytes.

The "root" filesystem houses the conventional Linux filesystem directories, system binaries and userspace applications. In order to contain a desktop environment, static analysis and programming tools, in addition to a regular Linux environment, the partition must be sized appropriately. As such, the partition has been sized at 5124 mebibytes.

A problem identified during the design and implementation phase was the process of how files could be saved to the USB for later analysis or acquisition; as StatOS is often run in RAM only mode, saving files to the USB would be impossible all file systems would reside in RAM. To resolve this issue, a fat32 filesystem, labelled "StatOS_data", is present to house acquired malware, suspicious files and other media to investigate. This partition can be expanded by the user to the desired size and mounted in the operating system in both RAM only mode and regular mode, allowing for users to move files between RAM and persistent storage when required.

Additional problems included that it would be difficult to transfer files into the USB; this was resolved through the aforementioned fat32 partition. As fat32 can be read by the majority of operating systems, the decision was made to alter portions of the boot process to copy any files present in the "StatOS_data" partition to the Desktop. This allowed users to view only the data partition of StatOS in other operating systems and natively transfer in files they deem to be suspicious, as shown in Fig. 8.

To support both the USB persistence and ramdisk requirements, a modified initial ram filesystem was used. As discussed, utilising a ramdisk filesystem considerably increases both the speed and forensic isolation of secondary storage. However, ramdisks incur a moderate to significant initial delay, whilst the operating system is copied into the ramdisk. Therefore, offering the ability to choose between persistence

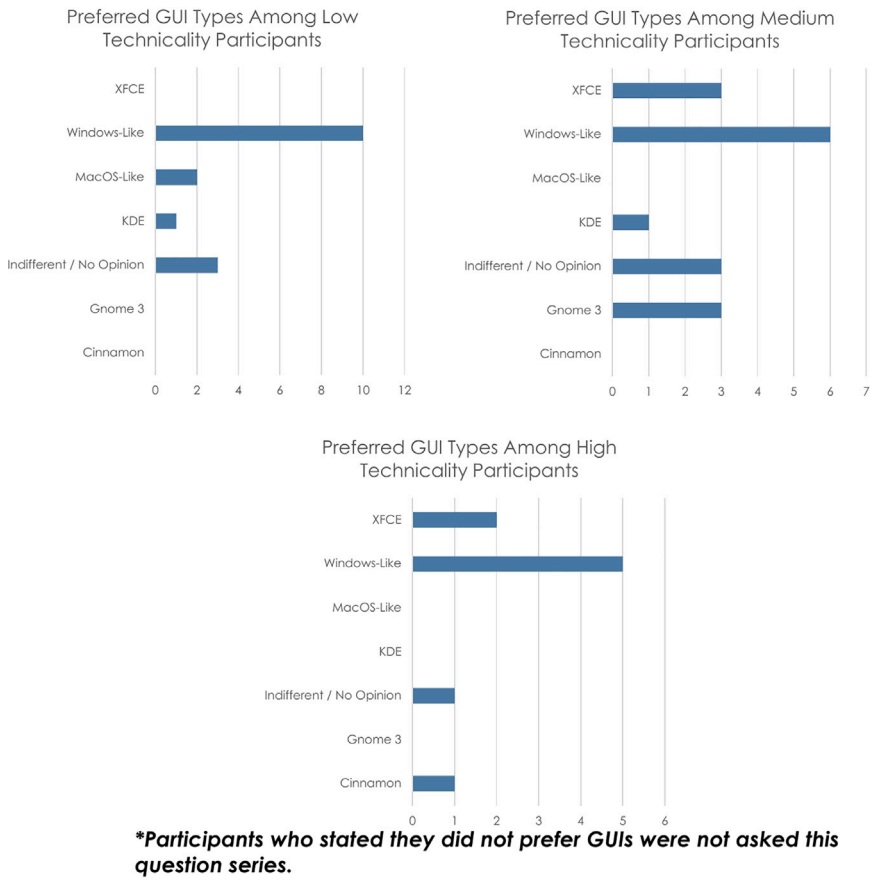

Fig. 5. Bar graphs communicating the most favoured GUI types in respondents who prefer GUI's.

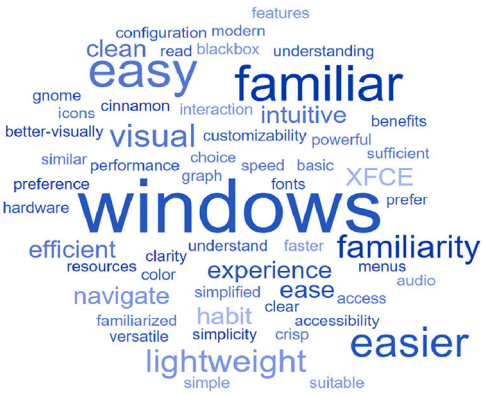

Fig. 6. A word cloud categorising all respondents' justifications for preferring GUI's, with words that occur more often enlarged.

or ram disk functionality at start-up gives a user greater control over how they wish to use the tool. However, launching with persistence and not utilising a ramdisk increases the risk of contamination of the USB device, as file artefacts could remain on the root partition of StatOS and persist across uses, instead of being confined to the dedicated "StatOS_data" partition. It is noted that launching with persistence, whilst increasing the risk of contamination, does have the benefit of circumventing time delays associated with copying into RAM, therefore allowing responders to enter the tool faster at the cost of heightened risk. If StatOS were to be used in training scenarios, testing, or if an individual wishes to persist changes, designing the operating system to support both possibilities would achieve far greater functionality.

To support both the USB persistence and ramdisk requirements, a modified initial ram filesystem was used. When executed, the initramfs would load into main memory and pause operation before mounting the "root" filesystem. At this point, a prompt was displayed to users, asking them to select if they wish to run from RAM, or run from the USB persistently. Depending on the selection, the initramfs would either generate a ramdisk and copy the contents of the "root" filesystem into main memory before switching root to this ramdisk, or mount the USB and proceed normally.

A notable issue encountered throughout the development of StatOS was ensuring that StatOS contained key software such as GUI's and analysis tools, whilst remaining small enough such that the entire operating system could be written to a USB drive with minimal storage. Whilst USB devices increasingly provide a reduced cost per gigabyte of storage, maintaining an operating system under four gigabytes allows for users to determine at boot time if the tool should copy itself into a ramdisk, or run from the USB device itself. Therefore, a smaller operating system footprint enables users the choice of ramdisk operation that may be otherwise unavailable.

This was achieved through removing features temporarily required during the compilation process, minimising the total size of the Linux kernel and through compiler options that minimise additional features deemed unnecessary or unrelated to the requirements of StatOS, as well as the exclusion of certain packages that were not absolutely necessary. Fig. 9 has been included to illustrate the result of these configurations.

5.1. OS GUI design

A key requirement of StatOS was the familiarity of a graphical user interface; this was achieved through the inclusion of XORG graphical user interface and supporting libraries. Based upon the preliminary

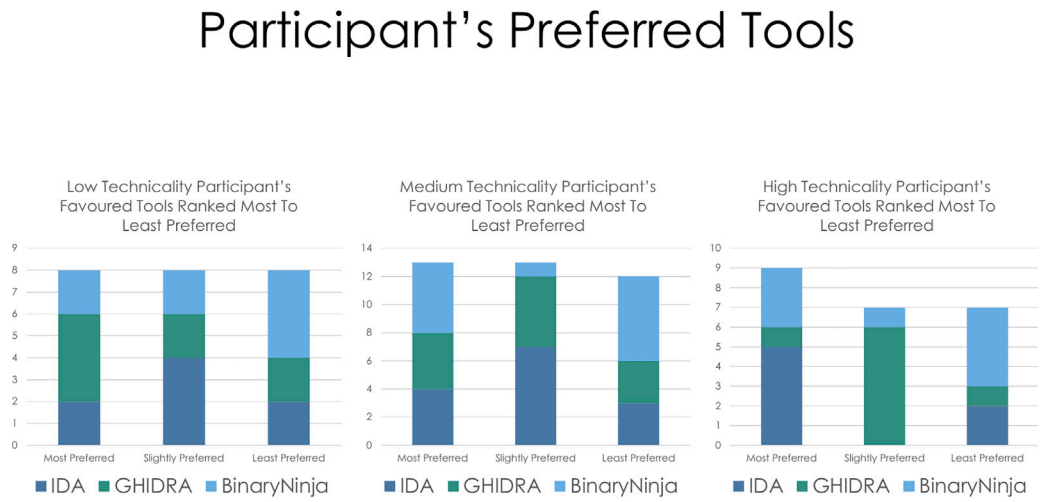

Fig. 7. Respondents opinions on favoured malware analysis tools.

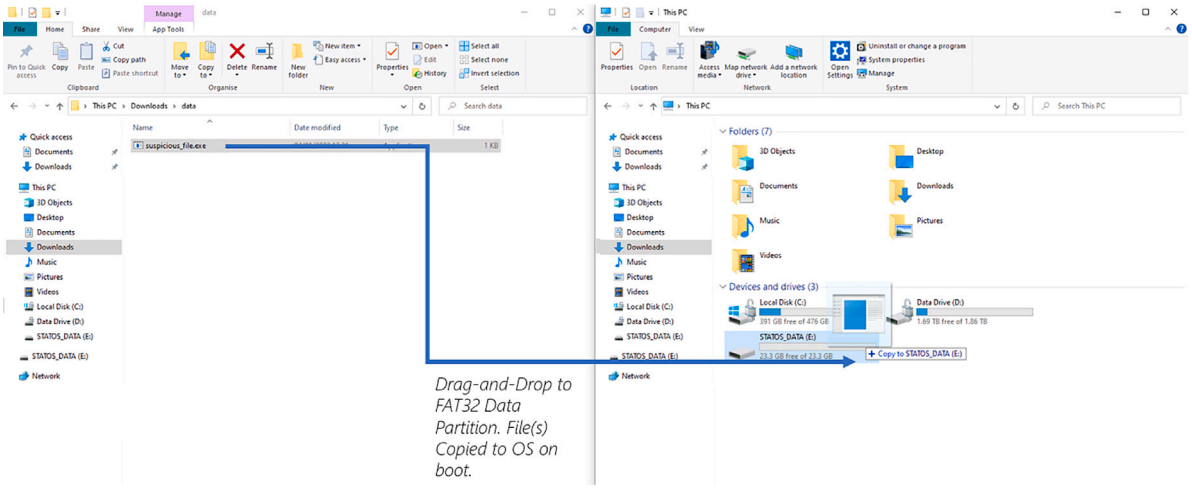

Fig. 8. A screenshot depicting the process of loading suspicious files into StatOS for analysis.

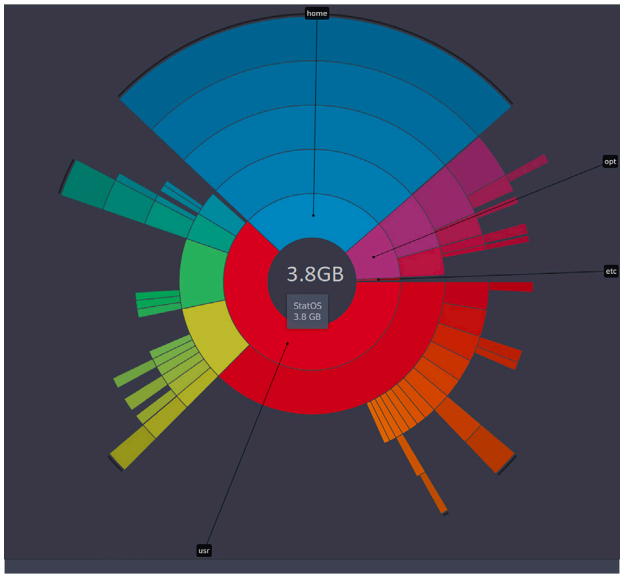

Fig. 9. A diagram revealing StatOS's OS size after package removals, kernel modifications and debug symbol removal.

survey displaying that respondents of all technicalities overwhelmingly prefer windows-like environments, the XFCE desktop environment was included. As XFCE is capable of emulating an environment similar to Microsoft Windows' GUI's whilst remaining relatively lightweight, the desktop environment was an extremely viable candidate.

To modernise the environment, XFCE4's "panel" functionality was used to create a border on the top of the GUI window, containing a start menu, application tray and system tray. From this menu, the operating systems core utilities and analysis tools could be operated. A secondary panel was used to create a middle-centred border on the bottom of the GUI, containing application launchers that are configured to point at the underlying binary executable files. The design decision was taken to include this secondary menu to clearly identify analysis tools upon entering the GUI in a user-friendly manner, whilst still retaining the windows-like top menu. An annotated image of the desktop can be seen in Fig. 10.

Two folders were created on the desktop; a tools folder and a folder named "Imported Artefacts". The purpose of the tools folder is to house the GHIDRA java tarball, such that a user could modify the installed version if required. The artefacts folder was designed to house suspicious files deposited by a user for investigation, as well as being populated during boot-time by the contents of the "StatOS_data" FAT32 partition. This would allow users to deposit files onto the FAT32 partition for investigation from a target computer system, then launch

Fig. 10. A screenshot taken from StatOS showing the XFCE4 desktop.

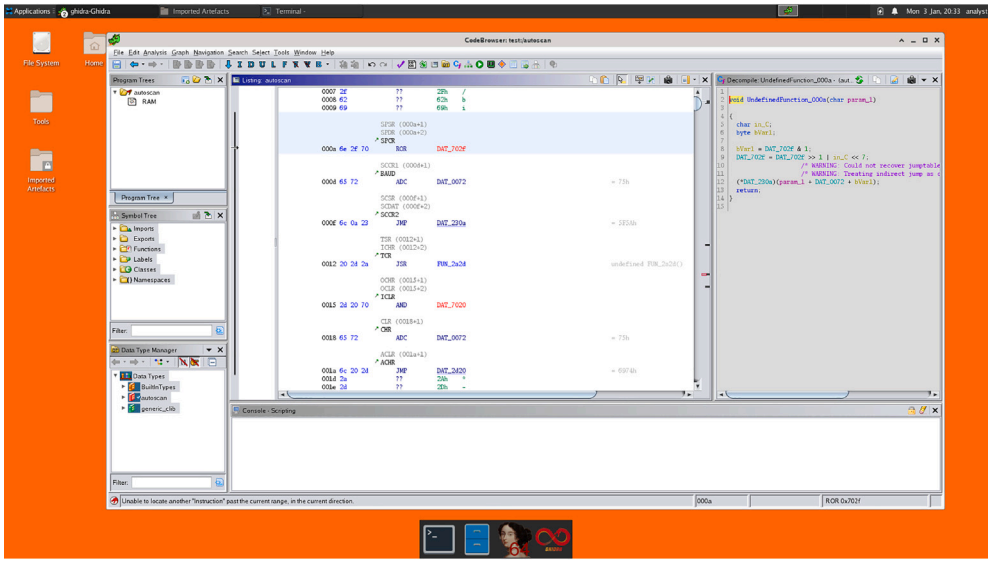

Fig. 11. A screenshot of GHIDRA being used to analyse a binary on StatOS.

StatOS in either ramdisk or persistent mode, at which point the contents of the partition would either be copied into RAM in the relevant folder or directly mounted respectively.

With the modern GUI and terminal interfaces in place, it was possible to install malware analysis tools such as IDA and GHIDRA, which were highly valued in low and high technicality groups during the preliminary survey results analysis. Programming tools such as Python and C compilers were additionally installed into the operating system and are accessible through both the GUI and terminal interfaces. An image of GHIDRA running on StatOS can be seen in Fig. 11.

5.2. Robustness assurances

To ensure StatOS is capable of presenting a high level of security and forensic isolation against accidental infection, a number of modifications are present within the design. A notable integrity feature is that a customised Linux kernel has been used, which has been compiled without the capability to interface with conventional secondary storage devices. Furthermore, only specific SATA drivers are included to support the kernel being able to identify and mount the USB disk itself, with the intention of preventing the device from being able to mount internal hard drives and potentially infect a host machine being utilised for analysis.

Defensive strategies such as the above are layered to provide a defence-in-depth approach [71]. In addition to removing kernel modules responsible for driving the majority of storage devices, the custom kernel has been stripped of its ability to interface with filesystems other than ext2, ext4, ramdisks and fat32 partitions, in an effort to further reduce the attack surface of StatOS and therefore the probability of an infection occurring [72]. Networking capabilities have additionally been removed where possible, to protect against the eventuality of a malware infection attempting to traverse a network.

To support this, the packages and their associated cryptographic checksums have been compared against the authoritative versions created by the package maintainers, to attest that the packages are legitimate copies [73]. Source code of the core operating system packages used to create the initial operating system will be validated through VirusTotal's sample submission analyser [74]. VirusTotal allows individuals and organisations to upload suspicious files, which

will be passed to a multitude of pattern-based, heuristic and artificial intelligence systems to identify potential malicious items [75].

Participants were not asked to comment on the aforementioned security controls. The researchers intention was to gain requirements and opinions to develop the potential tool, rather than ask participants their opinion on the specific security controls that should be deployed, as this would expand the scope of the initial research question considerably.

6. Integrity verification of StatOS source

To measure the integrity of StatOS, the core packages, libraries and source code responsible for creating the base operating system were analysed through VirusTotal, using the Monte Carlo methodology. To allow for multiple upload batches over time, automated upload scripts were utilised to manage the file-upload process and return the analysis links. Once the VirusTotal links were collated, a secondary Bash script was used to communicate with VirusTotal over a REST API to extract analysis results in JSON format.

In total, 83 source packages were analysed against 56 antimalware engine solutions, with VirusTotal performing repeated analysis daily over the course of one month with the intention of correlating new IOC's or malware against the source packages.

As highlighted above, 2/83 source code packages were identified by VirusTotal as potentially suspicious or harbouring malware. Investigating the results in further detail reveals the alerts were triggered by an archive within the tests directory of"e2fsprogs-1.46.4" and two shell scripts within the test directory of "util-linux-2.37.2". Both alerts were generated by the same antimalware engine with the same score, MAX; MAX is an antimalware solution capable of using heuristic analysis and machine learning to attempt to identify malware, which ultimately led to the detection.

The detection of both software packages was detected as medium-high, indicating a potential presence of malware. Upon inspecting each software package using VirusTotal Graph, a mechanism that allows relationships between indexed samples to be discovered, it was found that the e2fsprogs package was likely triggered due to malware calling the library to interface with filesystems on a target computer system. As e2fsprogs contains utilities for maintaining ext2, ext3 and ext4 filesystems, applications that interface with such filesystems will call utilities from e2fsprogs. Both legitimate and malicious application seeking to manipulate filesystems will use e2fsprogs. In short, the package itself was innocent, however due to a large volume of indexed malware calling the library, the machine learning algorithm categorised the package as malicious by association with known malicious malware that contained function calls to applications compiled within the e2fspackage.

Regarding the second detection, the alerted shell scripts from util-linux's test folder were isolated and investigated. Comparing the source code of the alerted "functions.sh" and "run.sh" to the authoritative GitHub directory for util-linux displayed two differences. A commented URL had been changed, in addition to a modification to one section of if/else logic to catch edge-cases. As these alterations relate to compile tests which are not included in StatOS, there is no security impact.

6.1. Elimination of outdated components

Each of the 83 source code packages was subjected to a search against the United States Government's National Vulnerability Database, with the resulting vulnerabilities being collated and visible in Fig. 12.

Out of the 83 source code packages, 15 potential vulnerabilities were identified. Tables detailing the number of vulnerabilities, their associated CVSS Score Group and the most frequent vulnerability types per CVSS Score group have been included in Tables 5 and 6.

As illustrated, most vulnerabilities associated with StatOS's environment fall into the medium-high categories. While the majority of these vulnerabilities have either no impact due to the removal of all but essential kernel modules or restricted permissions, a subset of the vulnerabilities have the ability to impact StatOS's integrity and must be considered. It is suggested that, while useful, a CVSS score itself should not inform the majority of a security decision, because the scoring algorithm does not take into account the exposure of a specific system to a vulnerability, only the severity if exploited [76].

Such vulnerabilities and scores have been included below. Three vulnerabilities listed below have the potential to cause memory corruption, denial-of-service attacks or potentially arbitrary code execution in legacy or weakened applications.

- CVE-2021-39537
- CVE-2022-23219
- CVE-2022-23218

These vulnerabilities offer attackers a viable method to attempt to compromise or otherwise disrupt the standard activities of an analyst using StatOS. While all of the vulnerabilities require local access due to StatOS having no networking capabilities, CVE-2022-23218 and CVE-2022-23219 could allow denial-of-service attacks to be carried out relatively trivially. Additionally, both of these vulnerabilities can allow applications compiled without stack protectors to be vulnerable against buffer overflow attacks, thereby leading to potential arbitrary code execution. At the time of developing StatOS, these vulnerabilities were not

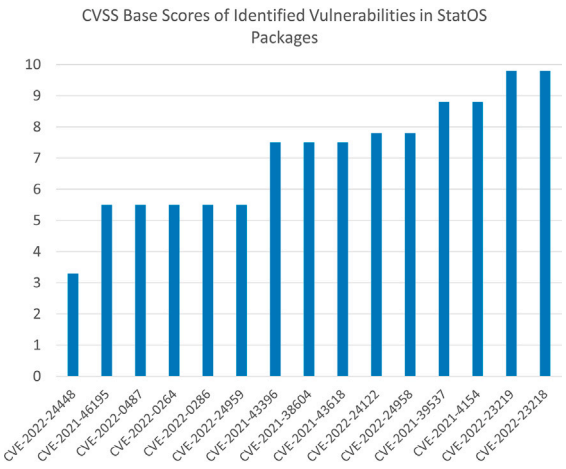

Fig. 12. A graph detailing each vulnerability and their associated CVSS score.

Table 5
A table of vulnerabilities in StatOS and their composite CVSS scores.

CVSS score group	Number of vulnerabilities
Low	1
Medium	5
High	7
Critical	2

Table 6
A table detailing the most common vulnerability types at each CVSS level.

Most frequent vulnerability type	CVSS score group
Missing Initialisation of Resource	Low
Mixed	Medium
Use After Free	High
Buffer Overflow	Critical

publicly known and therefore there was no mitigation technique that could have been implemented to protect against such vulnerabilities.

Based upon the analyses within Sections 5.2, 6 and 6.1, StatOS has largely achieved a suitable level of robustness whereby the design can be finalised and submitted for evaluation by experts within digital forensics and cyber security.

While there are vulnerabilities present within certain software packages that could lead to denial of service or potential buffer overflows if legacy software was used, StatOS is not intended to be installed onto computer systems and instead is intended to run temporarily to support field analysis and diagnostics, with the researchers mitigating these vulnerabilities once patches are available.

7. Delphi method interviews

To provide evaluative feedback of StatOS, 10% of the initial survey respondents were interviewed about the design and implementation. Interviews were semi-structured in nature, with questions being led through the use of a interview questionnaire to identify key areas to gain feedback. Through the use of remote video link software and in-person interviews, StatOS was demonstrated, with results being added to the survey as appropriate.

Towards the end of the interview, interviewees were asked directly about potential improvements and weaknesses within StatOS, if they had not already mentioned these subjects during the evaluation. An unstructured and brief conversation followed this to close the interview to collect any intangible or difficult to quantify sentiments or results.

During the software evaluation sections of the discussion, all respondents categorised the inclusion of GHIDRA and IDA as largely positive, with GHIDRA receiving the most positive responses. Persons testing StatOS found the tools useful for reverse engineering and static analysis, and found the performance benefits of running in RAM to be very useful.

In regards to evaluating OS performance, the boot time of 60 s to copy into RAM was met with mixed responses from different respondents, being classed as slow, moderate and fast by different respondents. By asking users what they believe an average amount of RAM is for average enterprise computer systems, it was possible to understand the feasibility of StatOS being used on existing equipment. Generally, 8 GB was considered moderate by respondents, which is a suitable amount of RAM to use StatOS and therefore, it can be implied that users believe StatOS is capable of running on moderately equipped hardware.

Further discussing boot times with users, it was asked if users would be willing to accept a longer boot time to initialise into RAM. While the majority of respondents agreed that this would be acceptable, Person D gave a counter argument, explaining that sometimes speed is required in forensic incidents and therefore investigations might not have time to wait for a complete initialisation, however stated that generally this depended on the severity of the incident being investigated.

Users were asked what they believed an average memory stick's size to be, in order to understand how effective StatOS could be mass-produced or deployed in a cost-effective manner. Such results indicated that sizes between 32–16 GB were considered moderate; as StatOS is capable of running from an 8 gigabyte USB, this reinforces the cost effectiveness and low hardware requirements of the solution. Users also generally believe that files being analysed are unlikely to exceed 1000 megabytes, which would be reaching the upper limitations of StatOS's current capabilities.

Regarding the initial requirements of StatOS, all users agreed that all the requirements discussed had been met, with respondents agreeing that portability and performance largely exceeded any requirements. A majority of users would use StatOS for forensic taskings, with potential future features such as signature-based malware detection and IDE's being suggested for inclusion. A table outlining these results is visible within Table 7.

7.1. Potential reservations

It is notable that of the 5 interviewees, whilst 4 would use a tool like StatOS within their organisation, one expressed reservations. The individual did not elaborate on their reasoning, however from the interview transcripts and responses to requirements relating to tooling, user-friendliness and terminal environments, the individual may have preferred an environment containing a greater selection of such tools, or potentially preferred to observe how StatOS would function over a prolonged time period.

As identified in Section 4.3, requirements relating to the development of StatOS were set out to definitively identify StatOS's strengths and weaknesses, as well as opportunities and threats. In the interest of further verifying requirements have been met, a convenient random sampling selection of industry experts who undertook the primary survey will give feedback on StatOS through semi-structured interviews and the delphi method. Through a combination of primary research surveys, Delphi-method interviews and Monte-Carlo source code analysis, StatOS has been created and evaluated. A final SWOT table illustrating StatOS's strengths, weaknesses, opportunities and threats has been included in Table 8.

8. Conclusions and future work

From the research and development undertaken, StatOS tool has been created in-line with the requirements sourced from individuals and experts within cyber security. The software was tested against such requirements through Monte-Carlo vulnerability analyses, before being evaluated through Delphi-method interviews with 10% of the initial survey respondents to provide confirmation of requirements being met.

From such research, it has been found that organisations often struggle to obtain budgeting for dedicated analysis hardware. StatOS would present a useful tool to mitigate against hardware shortages for themselves and their respective organisations, due to the low hardware requirements and relative ease of acquiring USB devices. Individuals found StatOS to meet all the requirements visible in Table 4, with StatOS exceeding requirements relating to portability and performance due to its ramdisk functionalities. These results indicate that StatOS represents a positive contribution towards the goal of alleviating hardware shortages in cyber security through the usage of Linux derivatives.

To enhance the viability and effectiveness of StatOS, future work is encouraged to develop a more comprehensive approach to package management, signature updating and the incorporation of vulnerability management processes into the development of the tool. Additional research should be undertaken with an increased sample size, as well as further practical use of StatOS to understand how individuals interact with the tool.

8.1. Limitations

Limitations include that integrity analysis was performed by the researcher; performing additional external research such as interviewing penetration testers would provide a more authoritative analysis source, in addition to potential threat modelling. Future research should seek to increase the overall number of participants to increase the reliability of the results, in addition to attempting to maintain StatOS over an extended period of time to understand its continued viability.

As mentioned within Section 3, whilst the sample size chosen is sufficient, a greater sample size would further strengthen the results. As such, the researchers recommend that future works attempt to gather a greater sample size across a wider geographical area and increase the number of participants, both at the survey and interview stages, to validate the results against a larger sample size.

Whilst collecting demographic information would provide a greater ability to analyse the data, the ethical implications of collecting such data would make timely data collection impractical, as well as requiring

Table 7
Delphi method results for StatOS.

ID	Requirements					Would use?
	Terminals and GUIs	Analysis and coding tools	User friendliness	Performance and portability	RAM and persistence	
1	Achieved	Exceeded	Achieved	Achieved	Exceeded	Yes
2	Achieved	Achieved	Achieved	Exceeded	Achieved	Yes
3	Achieved	Achieved	Achieved	Exceeded	Achieved	No
4	Exceeded	Achieved	Exceeded	Exceeded	Achieved	Yes
5	Achieved	Achieved	Achieved	Exceeded	Achieved	Yes

Table 8
Strengths, weaknesses, opportunities and threats (SWOT) analysis of StatOS.

Strengths	Weaknesses	Opportunities	Threats
Highly performant when running in RAM; capable of performing analyses rapidly.	Susceptible to physical attacks or attacks with physical access.	Updated software to mitigate CVE's.	Majority of analysis performed using tools such as SIFT/REMNUX/Kali/Parrot.
Highly portable due to USB3.0 device. Device is easily stored and transported, with minimal equipment required to operate.	Source compiled nature makes modifications difficult when deployed.	Include additional analysis software.	Upstream package management of other tools streamlines distribution and compatibility in a way which is not possible with StatOS.
Capable of switching between RAM modes for speed and persistence modes to save changes and investigation progress. Gives flexibility during analysis.	FAT32 data partition constrains analysis files to 4 GB maximum per file.	Inclusion of antimalware scanners.	Uncontrollable discovery of further CVE's being identified within software used.
Contains the two most requested analysis tools from the initial survey; GHIDA and IDA. Both are functional and can be operated effectively. Contains C compiler, Python2 and Python3 and shell scripting capabilities for programming.	Initial boot time is extended when copying into RAM.	Development of a reporting mechanism.	Alternative tools support additional architectures other than x86-64.
Provides both a command line user interface over TTY's and a GUI using XFCE4. Terminal emulation is possible in the GUI to allow both command line and GUI operation simultaneously.	Storage size limits the features provided by GUIs and CLI applications.	Potential support for dynamic analysis.	Greater development efforts and communities around existing tools ensure new features and software is integrated rapidly.
Feedback evidences that the user interface provided and the proposed mechanisms are user friendly to individuals within cyber security, who would be the primary target audience.	Limited sample size of respondents could be increased to afford greater reliability.	Further architecture support.	N/A
Can be quickly deployed for bulk analysis through USB device imaging, allowing blank USB devices to be converted into bootable StatOS devices	Requires technical knowledge to provision a StatOS USB or automate the process.	Potential for greater user friendliness through the integration of basic splash screens in initramfs.	N/A

substantial redaction, which would ultimately detract from the value of given demographic information [77,78]. Future research would benefit from collecting more comprehensive demographic information for greater analysis.

It is acknowledged that the key operating system features discussed in Section 4.1 and ranked by respondents within the preliminary survey were formulated from a review of existing literature rather than through direct primary research. Undertaking additional research with experts in digital forensics and cyber security could lead to the addition or alteration of these key features, which may ultimately improve the capabilities of StatOS.

It is recognised that both sampling methods chosen are non-probability sampling methods and, therefore, experience an elevated potential risk of bias compared to probability-based sampling methods. Whilst interviews and requirement sourcing with a larger selection of participants would be preferable, sample sizes smaller than the chosen sample in this study requirement sourcing have been demonstrated to be highly successful in requirement sourcing [79–81]. Additionally, probability-based sampling methods would require an extended time period to build a pool of research participants of similar size, which, whilst reducing the risk of bias, would detract considerably from the time available to develop and design the solution, leading to a solution with dramatically reduced capabilities.

To mitigate against such biases, respondents were selected from various organisational roles, across multiple organisations and in different industries to give a more comprehensive sample. Furthermore, the results included metrics on the percentage of individuals working at the

same organisation as the research organiser to address potential unconscious biases such as affinity bias. It has been suggested that Affinity bias occurs in individuals who feel an affinity or form of connection with another person; after working within related departments, it is possible that such a connection could be present, which could lead to an impact on the results [82]. By identifying such biases, it is anticipated that it will better reflect an accurate and true representation of the results [83].

Further mitigations include the research methodology decision to perform the preliminary survey before the design of the solution to gather insights. By performing the research in this manner, the risk of response bias in favour of the researcher was considerably diminished, as the respondents were not evaluating a solution and instead were giving experiences, insights and opinions. Further demographic information can be seen within Section 3, with additional discussion in Section 8.1.

CRediT authorship contribution statement

Alexander Cameron: Writing – original draft, Software, Investigation, Data curation, Conceptualization. **Abu Alam:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Formal analysis, Data curation. **Nasreen Anjum:** Writing – review & editing, Visualization, Validation, Resources, Project administration, Methodology, Formal analysis, Conceptualization. **Javed Ali Khan:** Writing – review & editing, Validation, Resources, Funding acquisition, Formal analysis. **Alexios Mylonas:** Writing – review & editing, Validation, Resources, Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.array.2025.100391.

Data availability

No data was used for the research described in the article.

References

[1] Anson S. Applied incident response. 1st ed. John Wiley & Sons; 2020, p. 3–21.

[2] Kleymenov A, Thabet A. Mastering malware analysis. 1st ed. Packt Publishing, Limited; 2019, p. 498–510.

[3] Anjum N, Yang Z, Saki H, Kiran M, Shikh-Bahaei M. Device-to-device (D2D) communication as a bootstrapping system in a wireless cellular network. IEEE Access 2019;7:6661–78.

[4] Karapoola S, Rebeiro C, Parekh U, Veezhinathan K. Towards identifying early indicators of a malware infection. In: Proceedings of the 2020 ACM Asia conference on computer and communications security. New York, NY, USA: ACM; 2020, p. 680–1. http://dx.doi.org/10.1145/3321705.3331006, URL https://dl.acm.org/doi/10.1145/3321705.3331006.

[5] Mell P, Kent K, Nusbaum J. Special publication 800-83 sponsored by the department of homeland security guide to malware incident prevention and handling recommendations of the national institute of standards and technology. 2005, p. 6–11. http://dx.doi.org/10.5555/2206294.

[6] Fang J, Yang Z, Anjum N, Hu Y, Asgari H, Shikh-Bahaei M. Secure intelligent reflecting surface assisted UAV communication networks. In: 2021 IEEE international conference on communications workshops. IEEE; 2021, p. 1–6.

[7] Sykosch A, Ohm M, Meier M. Hunting observable objects for indication of compromise. In: Proceedings of the 13th international conference on availability, reliability and security. New York, NY, USA: ACM; 2018, p. 1–8. http://dx.doi.org/10.1145/3230833.3233282, URL https://dl.acm.org/doi/10.1145/3230833.3233282.

[8] Caviglione L, Wendzel S, Mazurczyk W. The future of digital forensics: Challenges and the road ahead. IEEE Secur Priv 2017;15:13–5. http://dx.doi.org/10.1109/MSP.2017.4251117, URL http://ieeexplore.ieee.org/document/8123473/.

[9] Imran K, Anjum N, Alghamdi A, Shaikh A, Hamdi M, Mahfooz S. A secure and efficient cluster-based authentication scheme for internet of things (IoTs). CMC-Computers Materials & Continua 2022;70(1):1033–52.

[10] Schlette D, Caselli M, Pernul G. A comparative study on cyber threat intelligence: The security incident response perspective. IEEE Commun Surv & Tutorials 2021;23(4):2525–56. http://dx.doi.org/10.1109/COMST.2021.3117338.

[11] Nisioti A, Loukas G, Laszka A, Panaousis E. Data-driven decision support for optimizing cyber forensic investigations. IEEE Trans Inf Forensics Secur 2021;16:2397–412. http://dx.doi.org/10.1109/TIFS.2021.3054966.

[12] Makrakis GM, Kolias C, Kambourakis G, Rieger C, Benjamin J. Industrial and critical infrastructure security: Technical analysis of real-life security incidents. IEEE Access 2021;9:165295–325. http://dx.doi.org/10.1109/ACCESS.2021.3133348.

[13] Sun N, Zhang J, Rimba P, Gao S, Zhang LY, Xiang Y. Data-driven cybersecurity incident prediction: A survey. IEEE Commun Surv & Tutorials 2019;21(2):1744–72. http://dx.doi.org/10.1109/COMST.2018.2885561.

[14] Pliatsios D, Sarigiannidis P, Lagkas T, Sarigiannidis AG. A survey on SCADA systems: Secure protocols, incidents, threats and tactics. IEEE Commun Surv & Tutorials 2020;22(3):1942–76. http://dx.doi.org/10.1109/COMST.2020.2987688.

[15] Ozer M, Varlioglu S, Gonen B, Adewopo V, Elsayed N, Zengin S. Cloud incident response: Challenges and opportunities. In: 2020 international conference on computational science and computational intelligence. IEEE; 2020, p. 49–54.

[16] Attinasi MG, De Stefani R, Frohm E, Gunnella V, Koester G, Tóth M, et al. The semiconductor shortage and its implication for euro area trade, production and prices. Econ Bull Boxes 2021;4.

[17] Sikorski M, Honig A. Practical malware analysis. 1st ed. William Pollock; 2012, p. 465–75.

[18] Nagano Y, Uda R. Static analysis with paragraph vector for malware detection. In: Proceedings of the 11th international conference on ubiquitous information management and communication. New York, NY, USA: ACM; 2017, p. 1–7. http://dx.doi.org/10.1145/3022227.3022306, URL https://dl.acm.org/doi/10.1145/3022227.3022306.

[19] Berady A, Jaume M, Tong VVT, Guette G. From TTP to IoC: Advanced persistent graphs for threat hunting. IEEE Trans Netw Serv Manag 2021;18(2):1321–33.

[20] Chierzi V, Mercês F. Evolution of IoT linux malware: A mitre att&ck ttp based approach. In: 2021 APWG symposium on electronic crime research. IEEE; 2021, p. 1–11.

[21] D'Elia DC, Coppa E, Palmaro F, Cavallaro L. On the dissection of evasive malware. IEEE Trans Inf Forensics Secur 2020;15:2750–65.

[22] Northrop EE, Lipford HR. Exploring the usability of open source network forensic tools. In: Proceedings of the 2014 ACM workshop on security information workers, vol. 2014-November. New York, New York, USA: ACM Press; 2014, p. 1–8. http://dx.doi.org/10.1145/2663887.2663903, URL http://dl.acm.org/citation.cfm?doid=2663887.2663903.

[23] Zografopoulos I, Ospina J, Liu X, Konstantinou C. Cyber-physical energy systems security: Threat modeling, risk assessment, resources, metrics, and case studies. IEEE Access 2021;9:29775–818.

[24] Liu J, Yin T, Yue D, Karimi HR, Cao J. Event-based secure leader-following consensus control for multiagent systems with multiple cyber attacks. IEEE Trans Cybern 2020;51(1):162–73.

[25] Alves-Foss J, Song J. Function boundary detection in stripped binaries. In: Proceedings of the 35th annual computer security applications conference. New York, NY, USA: ACM; 2019, p. 84–96. http://dx.doi.org/10.1145/3359789.3359825, URL https://dl.acm.org/doi/10.1145/3359789.3359825.

[26] Rohleder R. Hands-on ghidra - A tutorial about the software reverse engineering framework. In: Proceedings of the 3rd ACM workshop on software protection. New York, New York, USA: ACM Press; 2019, p. 77–8. http://dx.doi.org/10.1145/3338503.3357725, URL http://dl.acm.org/citation.cfm?doid=3338503.3357725.

[27] Both JJ, Spaans P, Geana A, de Laat C. Analyzing and enhancing embedded software technologies on RISC-V64 using the Ghidra framework. 2020, p. 1–4.

[28] Bhat O, Yeprem Z, Lingesh V. Comparison of 3 reverse engineering tools emotet view project smart homes view project. 2019, p. 8–15. http://dx.doi.org/10.13140/RG.2.2.35123.07203, URL https://www.researchgate.net/publication/333907927.

[29] Ferguson J, Kaminsky D. Reverse engineering code with IDA Pro. 1st ed. Syngress Pub.; 2008, p. 5–18.

[30] Eagle C. The ida pro book, 2nd edition. 1st ed. No Starch Press; 2011, p. 29–34.

[31] Holzer A, Kinder J, Veith H. Using verification technology to specify and detect malware. 2007, p. 2–7.

[32] Maier D, Seidel L. JMPscare: Introspection for binary-only fuzzing. In: Proceedings 2021 workshop on binary analysis research. Reston, VA: Internet Society; 2021, p. 2–6. http://dx.doi.org/10.14722/bar.2021.23003, URL https://www.ndss-symposium.org/wp-content/uploads/bar2021_23003_paper.pdf.

[33] Elovici Y, Kachlon A, Kedma G, Guri M. AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies. 2014, p. 1–4.

[34] Win T-Y, Tianfield H, Mair Q. UCC 2014 : 2014 IEEE/ACM 7th international conference on utility and cloud computing : proceedings : 8-11 december, 2014, London, England, united kingdom. 2014, p. 1004–9.

[35] Kovacs F, Gcfa G. SANS institute information security reading room windows 10 as a forensic platform. 2021.

[36] Callegaro M, Manfreda KL, Vehovar V. Web survey methodology. Sage; 2015.

[37] Bano M, Zowghi D. A systematic review on the relationship between user involvement and system success. Inf Softw Technol 2015;58:148–69.

[38] Mao K, Capra L, Harman M, Jia Y. A survey of the use of crowdsourcing in software engineering. J Syst Softw 2017;126:57–84.

[39] LaToza TD, Van Der Hoek A. Crowdsourcing in software engineering: Models, motivations, and challenges. IEEE Softw 2015;33(1):74–80.

[40] Stol K-J, Fitzgerald B. Two's company, three's a crowd: a case study of crowdsourcing software development. In: Proceedings of the 36th international conference on software engineering. 2014, p. 187–98.

[41] Lei H, Ganjeizadeh F, Jayachandran PK, Ozcan P. A statistical analysis of the effects of Scrum and Kanban on software development projects. Robot Comput-Integr Manuf 2017;43:59–67.

[42] Li C, Huang L, Ge J, Luo B, Ng V. Automatically classifying user requests in crowdsourcing requirements engineering. J Syst Softw 2018;138:108–23.

[43] Flostrand A, Pitt L, Bridson S. The Delphi technique in forecasting–A 42-year bibliographic analysis (1975–2017). Technol Forecast Soc Change 2020;150:119773.

[44] McMillan SS, King M, Tully MP. How to use the nominal group and delphi techniques. Int J Clin Pharm 2016;38:655–62.

[45] Chowdhury N, Katsikas S, Gkioulos V. Modeling effective cybersecurity training frameworks: A delphi method-based study. Comput Secur 2022;113:102551.

[46] Dawood KA, Sharif KY, Ghani AA, Zulzalil H, Zaidan A, Zaidan B. Towards a unified criteria model for usability evaluation in the context of open source software based on a fuzzy Delphi method. Inf Softw Technol 2021;130:106453.

[47] Nugraha Y, Brown I, Sastrosubroto AS. An adaptive wideband delphi method to study state cyber-defence requirements. IEEE Trans Emerg Top Comput 2015;4(1):47–59.

[48] Hosseini M, Shahri A, Phalp K, Taylor J, Ali R. Crowdsourcing: A taxonomy and systematic mapping study. Comput Sci Rev 2015;17:43–69.

[49] Flostrand A. Finding the future: Crowdsourcing versus the Delphi technique. Bus Horiz 2017;60(2):229–36.

[50] Etikan I. Comparison of convenience sampling and purposive sampling. Am J Theor Appl Stat 2017;5:1. http://dx.doi.org/10.11648/j.ajtas.20160501.11, URL http://www.sciencepublishinggroup.com/journal/paperinfo?journalid=146&doi=10.11648/j.ajtas.20160501.11.

[51] Joshi A, Kale S, Chandel S, Pal D. Likert scale: Explored and explained. Br J Appl Sci Technol 2015;7:396–403. http://dx.doi.org/10.9734/BJAST/2015/14975, URL http://www.sciencedomain.org/abstract.php?iid=773&id=5&aid=8206.

[52] Saab F, Elhajj IH, Kayssi A, Chehab A. Modelling cognitive bias in crowdsourcing systems. Cogn Syst Res 2019;58:1–18.

[53] Wang J, Cui Q, Wang Q, Wang S. Towards effectively test report classification to assist crowdsourced testing. In: Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement. 2016, p. 1–10.

[54] Soprano M, Roitero K, La Barbera D, Ceolin D, Spina D, Mizzaro S, et al. The many dimensions of truthfulness: Crowdsourcing misinformation assessments on a multidimensional scale. Inf Process Manage 2021;58(6):102710.

[55] LeBreton JM, Ployhart RE, Ladd RT. A Monte Carlo comparison of relative importance methodologies. Organ Res Methods 2004;7:258–82. http://dx.doi.org/10.1177/1094428104266017.

[56] Allodi L, Banescu S, Femmer H, Beckers K. Identifying relevant information cues for vulnerability assessment using CVSS. In: CODASPY 2018 - Proceedings of the 8th ACM conference on data and application security and privacy, vol. 2018-January. Association for Computing Machinery, Inc; 2018, p. 119–26. http://dx.doi.org/10.1145/3176258.3176340, http://arxiv.org/abs/1803.07648.

[57] Scheibe M, Skutsch M, Schofer J. IV. C. Experiments in delphi methodology. Delphi Method: Tech Appl 2002;257–81.

[58] Primi R, Santos D, De Fruyt F, John OP. Comparison of classical and modern methods for measuring and correcting for acquiescence. Br J Math Stat Psychol 2019;72(3):447–65.

[59] Nyre-Yu M, Gutzwiller RS, Caldwell BS. Observing cyber security incident response: qualitative themes from field research. In: Proceedings of the human factors and ergonomics society annual meeting, vol. 63, no. 1. SAGE Publications Sage CA: Los Angeles, CA; 2019, p. 437–41.

[60] Guri M, Elovici Y. Bridgeware: The air-gap malware. Commun ACM 2018;61(4):74–82. http://dx.doi.org/10.1145/3177230.

[61] Jiroušek R, Kratochvíl V. Ambiguity effect: decision-making influenced by lack of information. In: 2021 IEEE international conference on technology and entrepreneurship. IEEE; 2021, p. 1–6.

[62] Talluri BC, Urai AE, Tsetsos K, Usher M, Donner TH. Confirmation bias through selective overweighting of choice-consistent evidence. Curr Biology 2018;28(19):3128–35.e8. http://dx.doi.org/10.1016/j.cub.2018.07.052, URL https://www.sciencedirect.com/science/article/pii/S0960982218309825.

[63] Votipka D, Punzalan MN, Rabin SM, Tausczik Y, Mazurek ML. An investigation of online reverse engineering community discussions in the context of ghidra. In: 2021 IEEE European symposium on security and privacy. IEEE; 2021, p. 1–20.

[64] Vaithilingam P, Guo PJ. Bespoke: Interactively synthesizing custom GUIs from command-line applications by demonstration. In: Proceedings of the 32nd annual ACM symposium on user interface software and technology. 2019, p. 563–76.

[65] Isaacs KE, Gamblin T. Preserving command line workflow for a package management system using ASCII DAG visualization. IEEE Trans Vis Comput Graphics 2019;25(9):2804–20. http://dx.doi.org/10.1109/TVCG.2018.2859974.

[66] Chi SS, Shanthikumar DM. Local bias in google search and the market response around earnings announcements. Account Rev 2017;92(4):115–43.

[67] Voronkov A, Martucci LA, Lindskog S. System administrators prefer command line interfaces, don't they? an exploratory study of firewall interfaces. In: Fifteenth symposium on usable privacy and security. 2019, p. 259–71.

[68] Tan X, Zhou M, Fitzgerald B. Scaling open source communities: An empirical study of the linux kernel. In: 2020 IEEE/ACM 42nd international conference on software engineering. IEEE; 2020, p. 1222–34.

[69] Lawall J, Muller G. Coccinelle: 10 years of automated evolution in the linux kernel. In: 2018 USeNIX annual technical conference. 2018, p. 601–14.

[70] Rangan CA, Holla KA, Kulkarniz V, Kumarx A, Patil A. Data rate based performance analysis and optimization of bulk OUT transactions in USB 3.0 SuperSpeed protocol. In: 2018 second international conference on advances in electronics, computers and communications. IEEE; 2018, p. 1–6.

[71] Maglaras LA, Kim K-H, Janicke H, Ferrag MA, Rallis S, Fragkou P, et al. Cyber security of critical infrastructures. ICT Express 2018;4(1):42–5. http://dx.doi.org/10.1016/j.icte.2018.02.001, URL https://www.sciencedirect.com/science/article/pii/S2405959517303880.

[72] Ghavamnia S, Palit T, Mishra S, Polychronakis M. Temporal system call specialization for attack surface reduction. In: 29th USeNIX security symposium. 2020, p. 1749–66.

[73] Hof B, Carle G. Software distribution transparency and auditability. 2017, arXiv preprint arXiv:1711.07278.

[74] Peng P, Yang L, Song L, Wang G. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In: Proceedings of the internet measurement conference. 2019, p. 478–85.

[75] Masri R, Aldwairi M. Automated malicious advertisement detection using VirusTotal, URLVoid, and TrendMicro. In: 2017 8th international conference on information and communication systems. 2017, p. 336–41. http://dx.doi.org/10.1109/IACS.2017.7921994.

[76] Spring J, Hatleback E, Householder A, Manion A, Shick D. Time to change the cvss? IEEE Secur Priv 2021;19(2):74–8. http://dx.doi.org/10.1109/MSEC.2020.3044475.

[77] Zanatta AL, Machado LS, Pereira GB, Prikladnicki R, Carmel E. Software crowdsourcing platforms. IEEE Softw 2016;33(6):112–6.

[78] Hirth M, Jacques J, Rodgers P, Scekic O, Wybrow M. Crowdsourcing technology to support academic research. In: Evaluation in the crowd. crowdsourcing and human-centered experiments: dagstuhl seminar 15481, Dagstuhl Castle, Germany, November 22–27, 2015, revised contributions. Springer; 2017, p. 70–95.

[79] Stolee KT, Elbaum S. Exploring the use of crowdsourcing to support empirical studies in software engineering. In: Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement. 2010, p. 1–4.

[80] Hosseini M, Shahri A, Phalp K, Taylor J, Ali R, Dalpiaz F. Configuring crowdsourcing for requirements elicitation. In: 2015 IEEE 9th international conference on research challenges in information science. IEEE; 2015, p. 133–8.

[81] Tsai W-T, Wu W, Huhns MN. Cloud-based software crowdsourcing. IEEE Internet Comput 2014;18(3):78–83. http://dx.doi.org/10.1109/MIC.2014.46.

[82] Gammie E. Unconscious bias and professional skepticism. Int Account Educ Stand Board (IAESB) 2018.

[83] McCormick H. The real effects of unconscious bias in the workplace. UNC Exec Dev Kenan- Flagler Bus Sch DIRECCIÓN 2015.