Green Security: A Framework for Measurement and Optimization of Energy Consumption of Cybersecurity Solutions

Sagi Brudni, Sapir Anidgar, Oleg Brodt, Dudu Mimran, Asaf Shabtai, Yuval Elovici
Telekom Innovation Labs at Ben-Gurion University of the Negev
Beer-Sheva, Israel
brudnis@post[.]bgu[.]ac[.]il, sapirani@post[.]bgu[.]ac[.]il, bolebg@bgu[.]ac[.]il,
dudu@dudumimran[.]com, shabtaia@bgu[.]ac[.]il, elovici@bgu[.]ac[.]il

***Abstract*—Information and communication technology (ICT) is playing an expanding and critical role in our modern lives. Due to its proliferation, ICT has a significant impact on global energy consumption, which in turn contributes to air pollution, climate change, water pollution, etc. The proliferation of ICT has been accompanied by the emergence of cybersecurity technologies and solutions, which play an integral role in society's digitalization. Wherever there is ICT, there is a need to secure it, resulting in an increase in global cybersecurity energy consumption as well. This paper discusses the energy-related aspects of cybersecurity solutions and defines a "Green Security" taxonomy. We highlight the inefficiencies stemming from various cybersecurity practices, such as processing the same data repeatedly. Within this context, we analyze cybersecurity solutions in common use cases, demonstrating the inherent energy consumption inefficiencies. In addition, we propose a method of measuring the energy consumed by cybersecurity solutions and present several optimization strategies that reduce their energy consumption. We evaluate our proposed optimization strategies and demonstrate their ability to reduce energy consumption while considering the organizational risk profile and maintaining the required security level.**

1. Introduction

Information and communication technology (ICT) is fundamental to the infrastructure of modern life. ICT enables us to share information, make video calls, send messages across the globe, pay online, consume software, and more. In fact, most of our critical infrastructure (including healthcare, energy generation and distribution, water supply, and communication infrastructure) and computer-enabled applications rely on ICT.

While important, ICT consumes a significant amount of energy, imposing a high cost on society. ICT's proliferation puts pressure on the energy supply, and it has become a contributing factor in climate change, [21], [24], [52]. With the increasing demand for computational power in the era of digitalization, energy consumption is expected to rise in order to meet the insatiable energy requirements of computation.

Currently, generating the energy essential for ICT (e.g., by combustion of fossil fuels) comes with an environmental cost in the form of climate change, air and water pollution, etc. [28], [68]. Ultimately, this environmental degradation negatively affects our health and the health of our planet [55]. According to the Intergovernmental Panel on Climate Change, global warming has caused the extinction of hundreds of species, the retreat of glaciers, reduced food and water security, harm to the physical and mental health of people around the world, and more [65]. According to [19], the portion of ICT in worldwide emissions is estimated to be between 2.1-3.9%, a figure that will not decrease without a concerted effort to make ICT "greener." In addition to its direct environmental and health benefits, ICT energy efficiency has clear financial benefits for organizations and society at large, such as decreasing the load on national and local power grids and reducing energy costs.

Various attempts have been made to reduce the energy toll of ICT. As the low-hanging fruit in ICT energy reduction, cloud data centers addressed high energy costs by relocating infrastructure to areas with cheaper energy sources and transferring costs to customers and end-users [78]. Since the cooling infrastructure is typically a data center's greatest energy consumer [18], Nordic countries have become attractive locations for new data centers [4]. We have even seen ICT equipment placed underwater for the same reason of reducing cooling costs [17].

Taking a more technologically challenging approach, ICT equipment vendors focused on optimization, developing more energy-efficient hardware; for example, GPU vendors have invested in producing more energy-efficient architectures [48], and there have been efforts to achieve silicon-level energy efficiency, such as reducing off-state leakage current in integrated circuits [37], [64], [66]. Additional optimization techniques such as dynamic halting of hardware components that are not currently in use and smart sleeping were reviewed in [16], [45]. In each of these methods, each improvement requires replacing old unoptimized hardware with new hardware. Depending on the use case, the expected hardware lifetime, and usage patterns, the energy savings achieved by hardware optimization can be offset by hardware replacement costs.

In recent years we have witnessed nascent efforts to invest in software-level energy efficiency, which entails the use of energy-aware operating systems, virtualization layers, and even individual software components [33]. Such efforts are of considerable significance due to their energy-saving potential and because replacing inefficient software components with energy-aware code is typically easier than replacing hardware.

The leading energy optimization efforts along with their benefits and drawbacks are summarized in Table 1.

DOI 10.1109/EuroSP60621.2024.00043

TABLE 1. A SUMMARY OF THE ADVANTAGES AND DISADVANTAGES OF THE LEADING OPTIMIZATION EFFORTS PRESENTED IN THE LITERATURE.

Optimization method	Description	Advantages	Disadvantages
Relocation of infrastructure	Relocating cloud data centers to cooler areas or places where energy is cheaper	Lower cooling costs, cheaper energy sources	Significant initial investment, backing up existing data is required
Hardware-level improvements	Enhancing the energy efficiency of hardware's architecture and management, e.g., reducing off-state leakage current	Application-agnostic optimization	Requires replacement of old hardware
Software-level improvements	Enhancing the energy efficiency of applications, OSs, virtualization, etc.	Easier than reallocating infrastructure and replacing hardware	In some cases, requires access to the source code

Despite these efforts, the overall energy consumption associated with ICT is still expected to increase, although the extent of this growth remains debatable. While some have argued that such growth in energy consumption will explode 15-fold (between 2010 and 2030) [8], others have asserted that the tidal wave of power demand will be completely offset by technological progress and advancements in green technologies [42].

Since energy efficiency is closely tied to green technological progress, we aim to shed light on a domain whose energy-saving potential has not yet been tapped – the cybersecurity domain. We focus on making cybersecurity solutions more energy efficient for several reasons. First, cybersecurity solutions are an essential part of ICT, and as such, they represent a portion of a technology's total energy footprint. In [21], the author proposed an assumption that 10% of all ICT equipment is devoted to cybersecurity, in addition to 20% of ICT system operations (such as attacks, patching, updates, upgrades, etc.) which are also devoted to cybersecurity. If this assumption holds true, then 10-20% of the total global ICT emissions are attributable to cybersecurity, resulting in annual CO_2 emissions of 142.5 million metric tons, which makes cybersecurity accountable for approximately 0.4% of the worldwide carbon emissions [21], [72]. These figures emphasize the importance of green security and the pressing need to formulate strategies aimed at reducing energy consumption in the cybersecurity realm. Second, most cybersecurity solutions are not energy aware or energy efficient, making them a perfect blue ocean candidate, where significant energy savings could potentially be obtained with little effort. Third, cybersecurity solutions are woven into every layer of ICT's technological stack – they can be found in hardware, operating systems, hypervisors, and even in individual software components. Fourth, the proliferation of cybersecurity tools in recent decades has resulted in a situation in which each computer and server in an organization usually runs several active cybersecurity solutions without consideration of the effect of this practice on energy consumption. Fifth, the predominant "layered security" defense strategy may lead to unnecessary redundancy of security controls, which taxes the overall security energy consumption, and finally, in many cases, increasing the energy efficiency of cybersecurity products can be accomplished simply by deploying a software update.

In this research we aimed to: (1) propose a method for measuring and predicting the energy consumption of cybersecurity solutions, ensuring that the security community has a tool for assessing the energy toll of these solutions; (2) propose several initial optimization schemes for common cybersecurity tools and architectures, a step that will potentially lead to further optimization research by the cybersecurity community; (3) raise awareness among the cybersecurity community of the fact that cybersecurity solutions can be energy-hungry and that this hunger can be optimized without necessarily impairing the security posture of an organization; and (4) contribute further to the efforts of researchers and practitioners in the field by providing a structured overview of the domain and devising a relevant taxonomy.

We note that any proposed optimization techniques should meet the following requirements:

- *Viability requirement:* There should be a non-negligible cumulative reduction in the energy consumed, where the total energy consumption of the optimization solution should be less than the energy saving that it achieves.
- *Security requirement:* The security level should remain unharmed when optimization measures are deployed. If that is not the case, the tradeoffs should be clearly understood and stated.

We start by defining a taxonomy of "Green Security" based on previous research in this domain. The taxonomy contains the main techniques used to measure and reduce the energy consumption of cybersecurity technologies. Where relevant, it also includes the tradeoffs between achieving higher levels of energy optimization and the resulting detrimental effects on the level of security.

Next, we analyze several use cases where the energy consumption of cybersecurity solutions is inefficient. We demonstrate that most of the inefficiency is rooted in repeatedly performing identical or similar security checks. This redundancy occurs by processing multiple copies of the same data across the organization and by performing similar cybersecurity checks by several different cybersecurity solutions.

Then, we present our method for measuring the energy consumed by any process on a computer system in general, and a cybersecurity solution in particular, based on resource consumption. The proposed measurement method enables separation of the energy consumed by a cybersecurity solution from the energy consumed by other processes on the same device. Our method assumes no prior knowledge of the inner-workings of the cybersecurity solution, treating it as a black-box. We use a dedicated Python program that tracks the hardware usage in a predefined time window (e.g., number of disk reads and writes, CPU and RAM usage) and tie this resource consumption to a specific process.

We also propose several ways to optimize cybersecurity solutions in terms of their energy consumption. We focus on optimizing the energy consumption of antiviruses while taking into account the associated risks. We demonstrate that while some of the proposed energy optimization methods do not have any impact on the security level, others do. Finally, we evaluate the performance of our proposed energy measurement method and optimizations. Our results indicate that non-negligible energy savings can be achieved by using simple optimization strategies.
To conclude, the main contributions of this paper are:

- a comprehensive review and first-of-its-kind taxonomy of green security;
- raising awareness among the cybersecurity community of the urgent need for green security;
- highlighting several scenarios where cybersecurity could be more efficient in terms of its energy consumption;
- a process-agnostic framework for estimating the energy consumption of each process, without relying on its source code; and
- energy optimizations for common cybersecurity solutions that consider the energy/security tradeoff.

2. Motivating Use Cases

In this section, we present some typical cybersecurity use cases that can be energy wasteful. The common ground for most of the waste is rooted in the redundancy of cybersecurity checks. If such redundancies do not contribute to the level of security, they are deemed unnecessary and therefore wasteful from an energy perspective.

2.1. Repetition of the Same Security Actions Across Different Endpoints Within the Organization

In this scenario, cybersecurity solutions, such as an antivirus, scan the same file - which exists across multiple machines in the same organization - even though the files are identical copies of one another. This is a common situation which may occur when a file is sent as an attachment to multiple employees in the organization. Since the files are identical, and the endpoint antivirus engine is usually the same across the enterprise fleet, it is not reasonable to assume that the scan results will differ across different machines.

Another example with the same result of redundant scans on different machines across the organizational fleet is when file sharing services are used within the enterprise, including cloud-based file-sharing solutions. When a file is created by a user and placed into a file share, it is replicated on the endpoints of the other users subscribed to the file sharing service. Such replication leads, yet again, to the performance of identical (redundant) antivirus scans. In other words, instead of scanning the file once, there will be n scans, where n is the number of file replications.

2.2. Repetition of the Same Security Actions on an Endpoint Over Time

A similar problem arises when the same file is checked repeatedly at different points in time. It makes sense for an antivirus to scan a file again in cases where the file was changed or when relevant new signatures have been added. However, it does not make sense to scan the same file again in cases in which it was not changed and there are no new relevant signatures in the antivirus' signature database. For example, if an organization runs a daily antivirus scan on a folder of PDF files and no malware is found, there is no need to run an additional scan unless a new signature pertaining to the PDF files has been added (in which case the entire folder should be scanned again) or a file in the folder was changed (in which case only that specific file should be scanned again).

2.3. Performance of Similar Security Actions by Overlapping Security Mechanisms

The "defense-in-depth" and "layered security" security approaches that have prevailed in recent decades advocate the deployment of several layers of security controls. This way, if an adversary is able to bypass one security control, an organization is not doomed, since there are additional controls on the way. While this approach has its merits, it also has an energy toll. Therefore, there is a need to find the right balance between the two.

We claim that in-depth defense can be achieved without compromising the level of effectiveness when reducing the unnecessary overlaps between the different security controls. Namely, it is unnecessary to deploy two overlapping security solutions that prevent the same type of exploitation in the name of in-depth defense, since they both will be bypassed in the same way by an adversary. Moreover, it has been shown that in some cases, overlaps may even impair the security posture [41], [73]. Therefore, to achieve an optimal security/energy balance, non-overlapping security solutions stacked in layers on top of one another are needed.

An example of overlapping security solutions would be a network IDS that reconstructs files that flow through it and runs an antivirus agent on them immediately before the file reaches the target system, which also runs an antivirus scan.

2.4. Performance of Security Actions with Inefficient Security Control

In this use case, it is assumed that there are several security controls of the same type that perform the same action (e.g., IDSs of different vendors). Although the action performed is the same, the energy profiles of the controls may differ. From a green security perspective, provided that the same level of security is achieved by both controls, it would be preferable to use an energy-efficient control rather than an inefficient one. It should be noted that prior research was unable to conclude that there is a correlation between the security level and energy consumption of an antivirus. Specifically, antiviruses that consumed more energy did not necessarily perform better than their low-energy counterparts and vice versa [56].

2.5. Performance of Gratuitous Security Actions

This use case involves security measures that do not improve the cybersecurity posture and are therefore unnec-

essary. They may be deployed by mistake, due to overreaction or out of convenience. For example, an organization may employ an encryption service on a communication link, encrypting all traffic indiscriminately although not all of the traffic needs to be encrypted.

3. Related Work

In this section, we first review the progress made in the domain of energy consumption measurement and the challenges associated with this domain. After providing a general review, we look at the progress and challenges in the cybersecurity context. Then, we provide an overview of recent related energy optimization efforts. Finally, we discuss energy/security tradeoffs, as reflected in contemporary academic literature.

3.1. Energy Consumption Measurement

The Intel Running Average Power Limit (RAPL) tool serves as the basis of several methods proposed to measure the energy consumption of a device or running process. This software tool's interface enables measurement of the energy consumed by the CPU and RAM when the device is operating. In [54], the authors used the RAPL tool to measure the energy consumption when solving various programming problems in different programming languages. However, this tool is limited to the Linux OS. In contrast, our measurement framework is generic and will work on any OS.

Another basic approach used to measure a device's energy consumption is to attach an external energy measurement apparatus to the device under test. An example of such an approach was presented in [39], where the authors tried to measure the energy consumption of different encryption algorithms deployed in wireless sensor networks, using an oscilloscope that they connected to the electrical circuit. Their measurement relies on the equation $E = V * I * t$, where E is the energy consumption of the device under test, V is voltage around the device, I is the electric current, and t is the device's runtime. The voltage around the device is known, and the running time is easy to compute. Using an oscilloscope, the authors measured the voltage drop around a resistor that they connected to the circuit. The current in the circuit was calculated using the equation $V = \frac{I}{R}$. The problem with this approach is that it measures the total energy consumption of the device under test without being able to differentiate between the energy consumed by cybersecurity processes and the energy consumed by other operations performed by the device.

In Appendix B.1.1, we elaborate on these approaches for assessing the energy consumption of cryptography algorithms. While exploring the energy consumption of various cryptography mechanisms in different setups is important, especially within the context of battery-powered IoT devices, our literature review showed that the topic of energy consumption of cybersecurity technologies outside the realm of encryption is largely underexplored. An overview of the limited work performed in this area is presented below.

The energy consumption of six popular antivirus programs that scan smartphone applications was measured by [56]. The measurements were performed using Appscope [75], which is software that evaluates the power consumption of each hardware component (e.g., CPU, display, wireless, GPS) per process on Android smartphones by analyzing system calls and inter-process communication. The power consumption measurements are expressed by Appscope as watts consumed per second. The results showed that for most antiviruses under test, energy consumption is strongly affected by the application's scan time (as confirmed by our results). While some antiviruses under test displayed a correlation between energy consumption and application size, others did not. In addition, for most antiviruses under test, a scan performed on a malicious application consumed less energy than a scan of a benign application, since those antiviruses raise an alert upon the first heuristic triggering, without applying additional checks and moving to the next application. It is important to note that the authors did not find a strong correlation between energy consumption and the quality of malware detection. We therefore conclude that antiviruses can optimize the energy consumption without compromising the level of security. While their measurement framework is limited to Android and calibrated specifically to the HTC Google Nexus One, we propose a more generic framework that is able to measure the energy consumption of cybersecurity solutions on any device, using any operating system, including Windows.

Additional works aimed at estimating the energy consumption of specific cybersecurity mechanisms (e.g., IDS) are reviewed in Appendix B.1.2.

3.2. Energy Consumption Optimization

In this subsection, we review the efforts made thus far in optimizing the energy consumption of cybersecurity solutions. The authors of [29] reviewed various approaches for reducing the energy consumption of cybersecurity solutions. In a leading approach called offloading [38], computational tasks are shifted from resource constrained devices (e.g., IoT devices) to other machines, e.g., shifting calculations from battery-limited mobile devices to servers in order to extend the devices' battery life. This approach does not reduce the overall power consumption; instead it shifts the consumption from one device to another, while incurring additional energy-related transfer costs.

An optimization method called online/offline security (described in [29]) typically applies to cryptographic schemes. This method shifts cybersecurity-related computations across time, where some are completed in advance (offline), and others are invoked in real time (online) and rely on pre-calculations computed in the offline stage. This method provides faster service and requires less computations in real time, when a security service is invoked. For instance, in the method proposed in [53], key stream bytes are computed in the offline phase and used in a subsequent online phase when encryption is applied to traffic. Similar to the previously mentioned method, in most cases, this optimization method does not reduce the overall power consumption but rather shifts the consumption from one time to another. An exception to this phenomena occurs in cases in which the online phase uses the results from the offline phase repeatedly, thereby reducing the total energy

consumption and not just transferring energy consumption to different time periods.

Unlike both optimization methods described above, our research emphasizes reduction rather than redistribution. Instead of merely redistributing energy consumption between devices or time windows, we focus on reducing the total energy footprint of cybersecurity solutions. Additional research on this topic is reviewed in Appendix B.2.

3.3. Energy/Security Tradeoffs

In some cases, energy saving comes at the cost of impaired security. In those cases, a high level of security negatively impacts the energy consumed by the security solution; accordingly, an energy toll would be imposed on an organization requiring a high level of security. We believe that organizations can be more energy efficient by performing an informed risk-benefit analysis according to their security needs. The following studies helped us lay the groundwork for such an approach.

The authors of [13] investigated the energy/security tradeoffs in malware detectors. Before execution, detectors ensure the integrity of the code within executable memory pages by comparing it to a whitelist of pre-approved code. It is possible to configure which types of code the detector should check; for example, the detector can be configured to check kernel code pages, root processes, or even all of the code on the system. Checking more types of code increases energy overhead. In addition, detector continuously check that all kernel data structures are valid and do not exhibit malicious abnormalities. When the frequency of those checks decreases, the total energy consumed by the mechanism decreases. However, decreasing the frequency of the checks benefits the attacker, providing more time windows for the attacker to infiltrate the system, perform malicious operations, and cover their tracks by restoring the system to a pristine state. Therefore, a "balanced profile" is one that achieves the desired risk profile – one that is effective against the vast majority of rootkits while maintaining reasonable energy overhead. Other studies that highlight the tradeoffs between energy and security are reviewed in Appendix B.3.

4. Proposed Taxonomy

Our proposed *green security taxonomy* is presented in Figure 1. The taxonomy includes five main categories: (1) cybersecurity controls; (2) methods for measuring the energy consumption of those controls; (3) methods for optimizing the energy consumption of those controls; (4) tradeoffs associated with the optimization methods; and (5) the scope, which defines the number of participating devices and cybersecurity mechanisms.

Our taxonomy is devised from a typical corporate on-premises setting perspective and can aid in measuring corporate energy expenditures. It can serve as a mental model for organizations that wish to reduce their energy footprint, enabling them to compare different security mechanisms by mapping the various solutions against the taxonomy. For this reason, we do not consider cloud-based architectures in this paper (although cloud service providers should be "green security" minded, and their corporate environment can be evaluated using the taxonomy). Similarly, any pre-deployment security mechanisms (such as DevSecOps-related code analysis tools) are out of the scope of the current research.

4.1. Enterprise Cybersecurity Mechanisms

Cybersecurity is comprised of various mechanisms, some of which can be standalone (e.g., antivirus), while others are combined with additional mechanisms into a single security solution (e.g., different cryptographical algorithms). Since we can measure and optimize the energy consumption based on an entire security solution, as well as its specific components, both are considered in our taxonomy. We divide the different security mechanisms into five main branches:

(i) Cryptography: In this branch, we consider the algorithms constituting the basic cryptographic building blocks, including symmetric encryption (e.g., DES, 3DES), asymmetric encryption (e.g., RSA, ELGamal), as well as hashing (e.g., MD5, SHA). This category also includes key exchange mechanisms (e.g., Diffie-Hellman), digital signatures (e.g., ECDSA), and message authentication codes (e.g., HMAC). In addition, we consider the various cryptographic protocols and implementations, such as authentication (e.g., Kerberos, RADIUS) and traffic security protocols (e.g., TLS, SSL, IPSec)

(ii) Endpoint agents: This branch deals with security software that runs on an endpoint machine, such as laptops, mobile devices, desktops, and server computer systems. It includes various antivirus solutions, host-based firewalls and IDSs, EDR, email scanners, and spam solutions, as well as local access control mechanisms.

(iii) Network mechanisms: This branch includes various mechanisms that are deployed on the network (as opposed to endpoint deployments), such as firewalls (e.g., classic 5-tuple, state-full, next-gen), intrusion detection and prevention systems, proxies (e.g., web, SOCKs, reverse proxies), and WAFs. The basis of all such solutions is that they perform some sort of traffic inspection on data in motion before it reaches its destination. In that sense, in this category we can also include security mechanisms enforced by network devices (e.g., ACLs enforced by routers on traffic passing through them, port security on switches, and DHCP snooping on layer 3 switches).

(iv) Aggregation and security management platforms: In this branch we include systems whose goal is to centralize, aggregate, store, analyze, manage, automate, share, and display cybersecurity-related data, information, intelligence, and insights in a central location. Within this category are SIEM, SOAR, security event management and incident response, and intelligence sharing systems, as well as centrally managed organizational identity and access systems. These systems collect and transmit massive amounts of data that is processed by queries and machine learning algorithms, making them non-negligible energy consumers [59].

(v) Generic: This branch includes cases where the subsequent measurements or optimizations are agnostic to the underlying security mechanism and can be applied regardless.

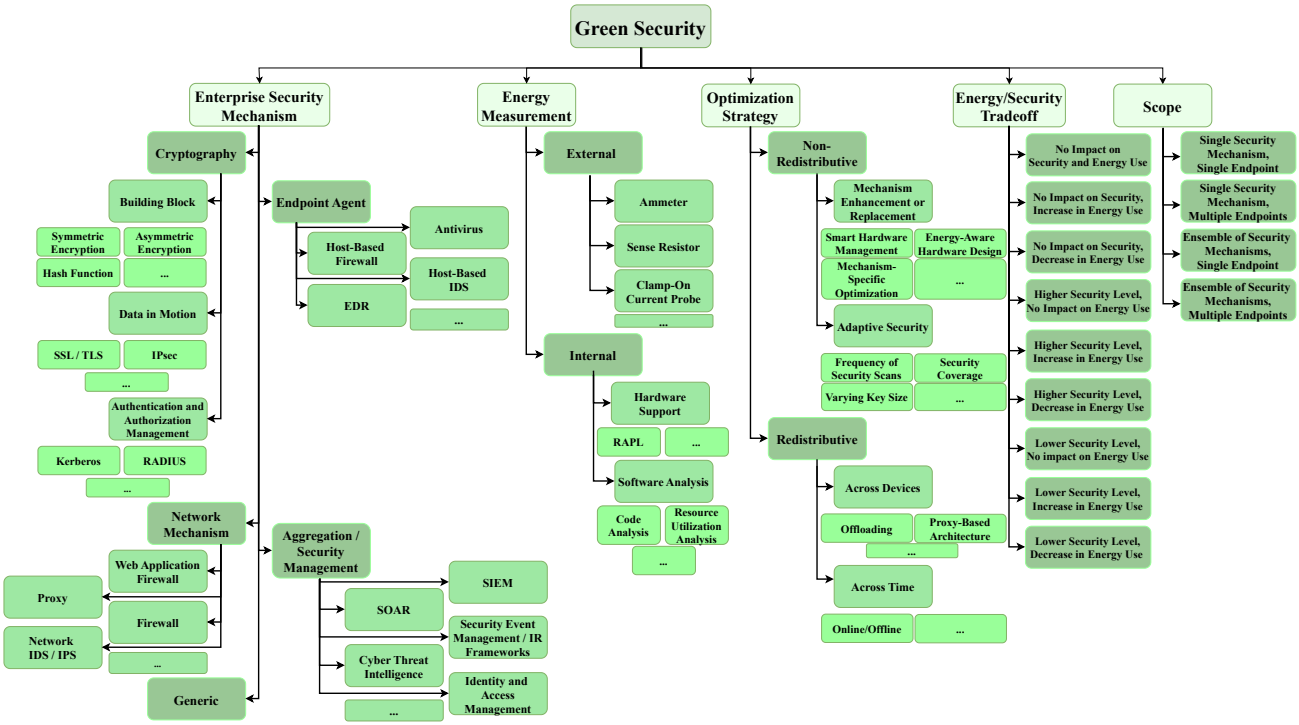

Figure 1. Proposed taxonomy for Green Security.

4.2. Energy Measurement

The next major node in the taxonomy is energy measurement, which deals with the different techniques by which the energy spending of the various security mechanisms described above can be measured. Since the energy consumption of a cybersecurity solution can be measured by using an external apparatus or an on-device technique, there are two branches in this category:

(i) External: This branch contains the various methods of measuring energy consumption using a measurement device that is external to the system. The use of an external energy measurement apparatus allows measurement of the energy consumed by the entire device under test, with limited ability to measure the energy consumption of individual processes. Typically, techniques in this branch strive to measure the current consumption (I) of the device under test when the voltage supplied by the power source (e.g., battery) (V) is known. The energy consumption is calculated by multiplying the measured current by the given voltage, while factoring in the device's runtime.

In the ammeter measurement technique (i.e., ampere meter), the current can be measured directly by attaching an ammeter in series to the power supply of the device under test; in the sense resistor technique, the voltage drop is measured (using dedicated equipment, such as an oscilloscope or voltmeter) across a sense (shunt) resistor that is attached in series to the power supply of the device under test. Given the resistance value, the current in the circuit is calculated using Ohm's law (as demonstrated in [39]). The limitation of both measurement techniques stems from the intervention in the electrical circuit required.

In contrast, the clamp-on current probe technique is non-invasive. This technique, as explained in [36], senses the strength of the electromagnetic field around the power supply conductor in the circuit, which is then converted to voltage by using a Hall effect sensor installed within the probe. After measuring the voltage output, the current in the circuit can be calculated easily based on the fact that the voltage at the output of the probe is proportional to the current. This technique was used in [13].

(ii) Internal: In this branch, the various measurement capabilities provided by the device under test itself are clustered, without the need for any additional external apparatus. Usually, these techniques enable a higher measurement granularity level than measuring the energy consumption of the entire device.

The first set of techniques in this branch leverages native hardware support for measurement, in which the hardware components are pre-equipped with dedicated sensors for energy consumption readings. An example of this approach is the RAPL technology available in Intel's processors [26].

The second set of techniques in this branch utilizes a software-related approach for monitoring. For example, as demonstrated in [25], an energy "price tag" for each assembly instruction running on a computer system can be produced. Then, we can analyze the software execution at runtime and monitor the assembly level instructions executed. By performing instruction-level inspection, the energy consumed by the executed program is estimated when considering the energy toll of each instruction.

An alternative software-based measurement that monitors the resource utilization of a specific process and estimates its energy consumption accordingly is presented in Section 5 and [75]. For example, we can write a program that tracks how much RAM and CPU a specific process utilizes and calculate the energy appetite of the process, assuming the energy consumption profile of CPU and RAM is known.

4.3. Optimization Strategy

The energy optimization category of the taxonomy deals with the various mechanisms by which the energy toll of cybersecurity solutions can be optimized. Energy optimization techniques are divided into two main branches. The first branch is non-redistributive, which strives to reduce the total energy consumption of cybersecurity or adjust it according to the required security level determined by an organization's policymakers. The second branch, redistributive, deals with redistributing the total energy consumption across devices or time, without necessarily reducing overall consumption.

(i) Non-redistributive optimization mechanisms: This branch includes techniques that deal with improving existing mechanisms or replacing them with more efficient mechanisms. For example, we can reduce energy consumption by designing energy-efficient hardware by various means, including reducing the off-state leakage current in integrated circuits (ICs) [66]. Another technique called smart hardware management is discussed in [45]. For example, by dynamically managing the idle intervals of an IDS according to the traffic load, energy savings can be achieved. Identifying a smarter ordering of the rules in firewalls and IDSs can also reduce the average processing effort of arriving packets.

Adaptive security is a sub-branch of non-distributive techniques. In this approach, instead of constantly providing the highest level of security possible, the overall energy consumption of security controls can be reduced by tailoring the required security level with a risk-based strategy. The risk can be evaluated by considering the sensitivity of the data and the potential threats [29].

The frequency of security scans is another aspect that can be adjusted to reduce the total energy consumption, again while considering the risk. Some cybersecurity solutions (e.g., antivirus) perform routine scans to identify cyber threats. When routine security scans are performed frequently, the attacker has a short time interval in between scans to infiltrate the system and execute malicious activities before being caught. However, high-frequency scanning comes with a high energy toll. This reflects the energy/security tradeoff, a tradeoff that must be considered when determining the scanning frequency (this tradeoff is covered in a separate branch in the taxonomy).

Another factor that affects the energy consumption of cybersecurity solutions is security coverage. The security coverage increases as the amount of data examined by the deployed cybersecurity solutions increases. Therefore, increasing security coverage typically requires greater energy consumption. For instance, the authors of [13] showed how the energy toll rises as the subsets of code pages and kernel data structures examined by rootkit detectors expand.

As stated in [39], [58], in the context of cryptographic algorithms, increasing the key size and the number of rounds improves the security level provided, but this comes at the cost of greater energy consumption. Adaptive security can leverage this phenomenon by dynamically adjusting cryptographic algorithms according to the sensitivity of the data to secure. For example, there may be a need to secure personal medical information sent over an untrusted network by using the largest possible key, while an email greeting sent to a colleague over a trusted network can be encrypted using a significantly shorter key or be sent unencrypted.

It should be noted, however, that adaptive security does not always result in reduced energy consumption. If an updated threat assessment necessitates a higher level of security (e.g., when a new threat actor targeting an organization's industry is identified or a new vulnerability is discovered in the critical system of an organization), the result could be increasing the level of security (temporarily or permanently), which may increase energy consumption even more rather than decreasing it.

(ii) Redistributive optimization mechanisms: Energy redistribution techniques do not necessarily reduce the total energy toll but rather deal with redistributing the demand. These techniques are especially important when cybersecurity meets resource-constrained devices, such as mobile phones and IoT devices which have limited battery life. The first category of redistributive techniques deals with energy redistribution across devices. A popular technique in this category is called offloading; in offloading, computations are shifted to another device in order to reduce the load on a constrained device, as demonstrated in [62]. Another common technique for redistributing energy leverages proxy-based architectures in which the client requests are transmitted to proxy servers instead of the requested destination server. In some cases, the proxy servers can provide an appropriate response back to the client without delegating the requests to the original destination server, thereby reducing its load. In the context of content delivery networks (CDNs), the proxy servers are physically located close to the clients and cache the responses from the main server, so that subsequent similar requests will be responded to rapidly by the nearest proxy server without being passed to the distant main server. In addition to improving the quality of service, this architecture results in redistribution of the energy load.

In the second category of redistributive techniques, the energy consumption is transferred across time. The online/offline technique shifts computations from the online phase to the offline phase; the offline phase occurs before the cybersecurity solution runs, while the execution of the cybersecurity solution takes place in the online phase. This approach was taken in [53], where the key of a stream cipher is generated in the offline phase and is only XORed later with the plaintext that arrives in the online phase.

4.4. Energy/Security Tradeoffs

This category covers the tradeoff between energy and security, allowing policymakers to consider the risk incurred by reducing the energy consumption of cybersecurity controls. While green security should strive to reduce the energy footprint of security mechanisms without impairing the level of security they provide, this is not always possible. To address this, we divided this category into nine possible outcomes that reflect the various combinations of the different security levels (increase/decrease/remain the same) and total energy consumption (increase/decrease/remain the same).

4.5. Scope

In this category, we define the scope of participating devices and cybersecurity mechanisms. For example, when devising an optimization method, we can consider the energy consumption of a single security mechanism running on a single device or that of an enterprise EDR solution as a whole, measuring the energy consumed over an entire fleet. We therefore include four possible cases in this category: (i) a single security mechanism that runs on a single device, without considering security operations performed on other devices; (ii) a single security mechanism that is deployed across several collaborating devices; (iii) an ensemble of security mechanisms that run on a single device, without considering security operations performed on other devices; and (iv) an ensemble of security mechanisms that are deployed across several collaborating devices.

The optimization proposed in Section 6.2 fits the first case (i.e., a single security mechanism on a single device), since we are monitoring the changes made to a local file and running a local antivirus accordingly. In contrast, optimization aimed at mitigating the energy wastage of an antivirus by avoiding redundant scans of files replicated across the organization (see Section 2.1) fits the second case. When optimizing different security agents running on the same endpoint, the cost-effective approach suggested in [14] fits the third case; the same optimization approach can fit the fourth case when optimizing the integration of multiple cybersecurity mechanisms (such as NIDS, EDRs, antiviruses, and firewalls) deployed across an organization.

5. Measurement Framework

In this section, we present our proposed framework for measuring the energy consumed by a running process.

We created our framework by performing steps i-iii. Step iv describes the use of our framework: *(i) Energy/resource measurement* - measure the resource utilization and its corresponding energy consumption for a wide variety of resource use combinations (e.g., number of disk read and write operations, CPU and RAM use, and the related energy consumption); *(ii) Dataset* - build a labeled dataset consisting of the measurements obtained in the previous step; *(iii) Prediction model* - use the dataset to build a machine learning model that predicts the energy consumption of any resource utilization combination; *(iv) Framework use* - monitor the resource utilization of a specific process of interest and apply the model built in the previous step to predict its energy consumption in a specific context (e.g., on specific hardware). The architecture and steps of our framework are presented in Figure 2, and a description of each step is provided below.

Our measurement framework is process-agnostic (the process for which the energy consumption is measured is treated as a black-box, meaning that the parameters and actions of the process are unknown when the energy is measured); the fact that it is process-agnostic means that it is generic, robust, extendable, and adaptable to different environments and platforms (e.g., hardware, OS).

5.1. Energy/Resource Measurement

The first step of our framework is aimed at creating a matrix of measurements that reflect the energy consumption of a computer system as a function of resource utilization. To accomplish this, we loaded different resources (e.g., CPU, I/O) with different load targets (e.g., CPU=10% and I/O=20 Mbps) and measured the corresponding energy consumption. Our methodology enables measurements from both laptop and desktop platforms to be obtained. Due to physical differences in the energy supply of laptops and desktops, obtaining the measurements from laptop systems is different than obtaining them from desktop systems.

Laptops: In battery-powered devices, we measured the energy consumption of the entire computer system based on the energy depleted from the device's battery.

Desktops: In order to measure the total energy consumed by stationary devices (non-battery-powered devices), we used an external device that measures the voltage and current passing to a desktop computer system when the measurement is obtained. The samples were taken frequently, and a current-voltage graph was constructed. First we integrated the current-voltage graph to obtain the power consumption during the measurement. Then, we multiplied the result by the measurement's duration to calculate the energy consumed by the device under test.

5.1.1. Measurement Setup. In this subsection, we present the measurement software that we developed, which is designed to measure the resource usage of every process that runs on the examined device on a granular level. The software is a generic Python program [1] that relies on an open-source library, psutil 5.9.4 [60]. We also added the functionality of reading the battery capacity for battery-powered devices using the WMI library [22] for Windows systems and the upower command [77] for Linux systems. By using the battery capacity readings, we could assess the energy consumption of laptops. In desktop devices, we used the program to read the internal resource utilization, while utilizing the external device to measure the energy consumption of the entire device under test, as explained above.

Our measurement software allows us to measure the baseline activity of the operating system when the device is in idle state, meaning that no additional user process runs on top. This configuration, along with the ability to measure the resources consumed by a specific process, makes it possible to determine the energy overhead of the process in question in terms of its energy vis-à-vis resource consumption. To obtain the measurements, we first loaded the system with a specific resource load profile. Once loaded, we measured the corresponding energy consumption (we also obtained a more granular measurement of the actual resource load). Then, we deducted the total energy consumption in an idle state from subsequent process measurements, in order to obtain the process-related consumption. That way, we can track the resources utilized by the process in question during a specific time window to assess the energy consumed by a process separately from the energy consumed by other processes running on the device.

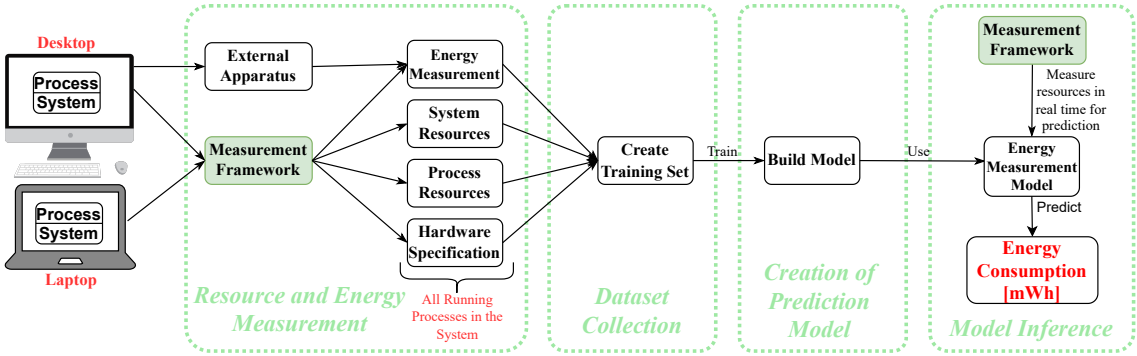

Figure 2. Proposed measurement framework, enabling quantification of the energy consumption of a running process.

5.1.2. Generate Resource Loads. We used the HeavyLoad program [69], which enabled us to set target load goals for various computer components. HeavyLoad also allowed us to generate processes and control their resource utilization profile; for example, we could generate a CPU-consuming process, a RAM-consuming process, and an I/O-consuming process. Using this program, we generated hundreds of different load combinations.

5.2. Dataset Creation

The measurements obtained in the previous step were saved in a dedicated labeled dataset which was used to train a machine learning model capable of predicting the energy consumption for any load profile. The dataset records include the resources used by a specific process, the resources used by the entire device when the measurement was obtained, and the hardware specification of the device executing the process. The resources used by the process and the entire device included in the dataset are: the CPU, memory, number of disk I/O reads, number of disk I/O writes, number of bytes of disk I/O read, number of bytes of disk I/O write, and number of page faults, as well as the corresponding energy used by the process (which is calculated by subtracting the total energy consumption in an idle state from the energy consumed by the device when it runs the specified process).

5.3. Prediction Model

The dataset is used to train a machine learning model capable of predicting the energy consumption of a process based on its resource consumption.

We divided the collected dataset into a training and test set at a ratio of 80-20%. In order to train the model, we measured the energy consumption of a variety of resource load profiles running on several devices individually.

To choose the best regression model, we trained and tested several regressors. After testing several models, we concluded that a random forest regressor provided the best results in most cases. Detailed explanations and the results of the various examined models can be found in Appendix C. It should be noted that due to the limited size of the dataset, the success of the models was limited; nevertheless it still demonstrates the viability of our approach, which could be expanded with more data in future research.

5.4. Framework Usage

In this subsection, we describe how users can use our proposed measurement framework to measure the energy consumption of a process of interest: (i) use the proposed measurement framework to measure both the resource use of the process in question as well as the entire system, while the process is running; (ii) use the collected data and the hardware specifications of the device running the process as input to the framework's machine learning model; and (iii) run the model to obtain a prediction of the energy consumption of the process.

To obtain the total energy consumption of the specific process across an organizational fleet, each prediction is multiplied by the number of devices with the same hardware specification; then the results are summed, as shown in Equation 1. An organization can estimate the energy spent in a specific domain (such as cybersecurity) by repeating the computations presented above for each process in the domain and summing the results, as shown in Equation 2. The model can also be used to calculate the total energy consumption of the entire organization, using Equation 3; in this case, the process of calculating the energy consumption of a specific domain is repeated for each domain in the organization.

$$EnergyConsumption_P = \sum_{c \in eachMakeAndModel} E_{p,c} n_c \quad (1)$$

$$EnergyConsumption_D = \sum_{p \in processesInD} E_p \quad (2)$$

$$EnergyConsumption_O = \sum_{d \in domainsInO} E_d \quad (3)$$

Equation 1 presents the calculations needed to calculate the amount of energy consumed by the organization when running the process p, where $E_{p,c}$ is the model's energy prediction for process p when it is run on the hardware of computer c, and n_c is the number of devices in the organization that have the same hardware specification as c. E_p in Equation 2 denotes the energy consumed by a process p that belongs to domain D. E_d in Equation 3 denotes the energy consumed by a domain d which is one of the O's domains (O is a typical organization).

5.5. Framework's Advantages

Our proposed framework has several advantages. First, no external hardware is needed to measure the energy consumption of a running process, since the energy consumption is predicted by the model based on resource utilization, which is measured by our measurement software, while taking the hardware specification of the device executing the process into account.

Second, our framework is generic and platform-agnostic, allowing it to be integrated into Windows and Linux-based systems, as well as both stationary devices and laptops.

In addition, our framework can receive input regarding multiple processes running on different platforms or running simultaneously on the same device. Then, based on the collected data, it can predict the energy consumption of each process individually. This feature makes our model flexible to changes in the environment, especially when an analysis of the energy consumption of several processes is required. Put differently, in order to measure the energy consumption of n processes individually, there is usually a need to first measure the energy consumed by the computer when executing each process individually and then compare the results to the total energy consumption in an idle state, necessitating obtaining $n+1$ measurements. However, our model can estimate the energy consumption of all n processes by performing a single measurement of the resource consumption of all of the processes at once.

By using the proposed measurement framework, an application designer can predict the energy consumption of an application before implementing it, based on the resources to be utilized by the application. An organization can also define energy-efficient policies and predict the application-related energy costs for each relevant domain (such as cybersecurity and data management) as we demonstrate, thereby reducing its energy use and costs.

5.6. Framework's Limitation

5.6.1. Scalability. In the context of measuring and evaluating the energy consumption of security countermeasures, our goal was to develop a machine learning-based approach for estimating the energy consumption, based on parameters that are relatively easy to obtain from the machine. However, to create a more robust and accurate model, measuring the energy consumption of machines at an enterprise scale involving hundreds or thousands of potentially diverse machines may be challenging and not always possible, as it requires the collection of data from different machines with different configurations. Therefore, the proposed method can suffer scalability challenges. Below are several strategies that may help alleviate some of these scaling challenges. Where relevant, we suggest approaching the scalability challenge by measuring the energy consumption of processes that run on identical hardware with identical configuration across the enterprise, relying on the fact that these processes consume similar hardware resources across devices and consequently have a similar energy footprint. We expect that some large enterprises will have hardware similarities, since for efficiency and cost, procurement in such environments may involve batch purchases of large volumes of similar items of the same (or similar) make and model thereby creating clusters of identical machines originating from the same vendor. This expectation is based on the fact that modern procurement strategies in large organizations advocate leveraging buying power to negotiate better bulk pricing and discounts [34]. This often leads to bulk buying in high volumes from a limited number of vendors who apply a "total quantity discount policy" pursuant to which the unit cost of a product decreases as the total quantity purchased increases [23], thus homogenizing the organizational IT landscape. Our framework may be beneficial in this setting, since in such cases the measurement can be limited to a representative machine in each such cluster and multiplied by the number of machines in each cluster. In organizations lacking such uniformity, the measurement tool can be deployed on target machines with common IT tools (e.g., pushing the tool as a part of a domain group policy [46]) or security tools typically used to push and run binaries on target machines during incident response (e.g., Kansa) [31] provided that organizational policy permits this. The tool would gather data and report back to a centralized server. In other cases our proposed approach's scalability may be limited, and it is our hope that the field of green security will evolve considerably, encouraging security vendors to provide data on energy consumption, thereby rendering our model unnecessary for measuring cybersecurity related consumption.

5.6.2. Security Management. We envision that green security will become an additional parameter to consider when designing, deploying, and applying security solutions. This raises various questions, such as integration with existing security frameworks, calculation of the direct financial benefits, ROI, and total cost of ownership (TCO), assigning a weight to green security considerations in risk-based decision-making, and determining the indirect benefits related to the reduction of environmental externalities. Although we do not have all the answers yet, we believe that consistent with the "shift-left" movement in cybersecurity, green security considerations should be factored in from the very beginning of the security lifecycle. Security leaders should start asking vendors green security questions, as doing so may increase the pressure on them to become greener. Likewise, new security solutions should be assessed in terms of the security they provide as well as the energy they consume. This will also allow more accurate OPEX calculations, since after all, you compare the security you obtain to the TCO of the control, which should also account for its energy demand.

6. Energy Optimization Strategies

In this section, we present several optimization strategies aimed at reducing the energy consumption of cybersecurity solutions. The first three optimizations are tailored to antivirus software, while the fourth can be applied on any end-point cybersecurity solution, considering it as a black-box. Namely, the latter only affects the scheduling of cybersecurity solutions, leaving their core logic unaltered. Before delving into the optimization strategies themselves, we discuss the need for such strategies.

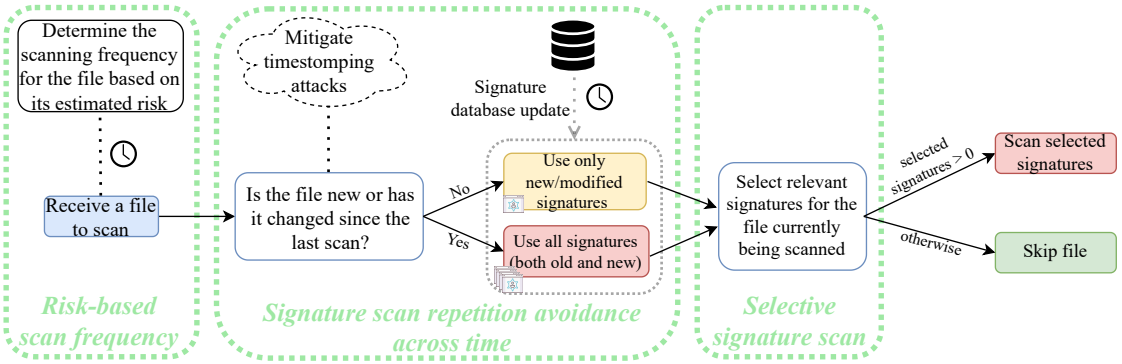

Figure 3. High-level optimization approach for antivirus.

6.1. Motivation for Antivirus Optimizations

As alluded to in Section 4, endpoint security solutions represent a significant portion of the overall security arsenal of a typical organization.

Due to their popularity and abundance (e.g., almost every laptop, workstation, and server runs one or more of these agents), they impose a non-negligible energy toll on organizations. We estimate that the annual financial costs of energy usage associated with running a typical antivirus solution alone across the fleet of a typical large US-based organization can amount to hundreds of thousands of US dollars (not to mention the energy costs associated with other endpoint security tools employed).

We base this calculation on an organization that employs $20,000$ employees (denoted as N_{emp}), where each employee has a desktop computer with the Microsoft Defender antivirus installed. Below we provide several antivirus usage scenarios, each relying on a different set of assumptions regarding the average amount of data on a typical desktop computer in such an organization and the number of antivirus scans performed per month.

In the most conservative scenario, we assume that an average computer hosts 100 GB of files (considering that the latest versions of Windows OS take almost 20 GB of disk space), and the organization runs a weekly scan. In this case, the associated financial expenses associated with antivirus energy consumption amount to approximately USD 5,000 annually. In the most extreme scenario, on the other hand, we assume that an average computer hosts 1.5 TB of files, and the organization runs a daily scan. In this case, the associated financial expenses associated with antivirus energy consumption come to approximately USD 570,000. Such a 100-fold variance in costs shows that antivirus scanning can incur significantly different costs in different organizations and that organizations should be aware that their scanning strategy has a direct impact on organizational energy consumption and the resulting financial costs. The total volume of data scanned annually by an antivirus is calculated as follows:

$$DV_{annually} = N_{emp} * DV^{w}_{avg} * N^{month}_{scans} * 12$$

where DV^{w}_{avg} denotes the average volume of data to be scanned per workstation and N^{month}_{scans} denotes the number of monthly antivirus scans; while the estimation of annual energy consumption attributed to full antivirus scans (in which all files of the device are scanned, typically at constant time intervals) is calculated as follows:

$$E^{antivirus}_{annually} = DV_{annually} * E^{typical}_{1MB}$$

As we show in Appendix D.1, the amount of energy consumed by an antivirus depends on the type of data being scanned. For simplicity in our calculation and based on empirical experiments, we assume here that the energy required by Microsoft Defender to scan 1 MB of "average" data is:

$$E^{typical}_{1MB} = 0.331mWh$$

We also assume a fixed price of 0.16 cents per kWh [50].

In Table 2 we present the estimated annual energy consumption associated with full antivirus scans based on various assumptions.

6.2. Antivirus Energy Optimization Strategies

In this subsection, we focus our analysis on antivirus energy optimization, but a similar analysis can also performed on other endpoint cybersecurity solutions.

Figure 3 presents our proposed antivirus energy optimization strategies which consist of three core components that can work in conjunction with one another: (i) risk-based scan frequency; (ii) signature scan repetition avoidance over time; and (iii) selective signature scan.

The optimization strategies are described below (the relevant background can be found in Appendix A).

TABLE 2. ESTIMATED ANNUAL ENERGY CONSUMPTION AND ASSOCIATED FINANCIAL OUTLAY OF AN ORGANIZATION COMPRISING 20,000 EMPLOYEES. EACH ROW REPRESENTS DISTINCT ASSUMPTIONS ACCORDING TO THE AVERAGE VOLUME OF DATA TO BE SCANNED BY THE ANTIVIRUS PER WORKSTATION (DV^{w}_{avg}) IN THE ORGANIZATION AND THE NUMBER OF FULL SCANS PERFORMED EACH MONTH (N^{month}_{scans}).

DV^{w}_{avg} **(TB)**	N^{month}_{scans}	**Energy Consumption (MWh)**	**Financial Expenses (USD)**
0.1	4	32	5,084
0.5	3	119	19,066
1	4	318	50,842
0.75	10	596	95,328
1.5	30	3,575	571,968

6.2.1. Risk-Based Scan Frequency. This optimization strategy is mainly suitable for full antivirus scans, in which all files of the device are scanned, typically at constant time intervals. Instead of determining a global scan frequency that is uniform for all files, we propose adjusting the scan frequency for each file based on its estimated risk. Intuitively, the more likely a file would be infected with malware and the greater the potential of resulting damage, the more frequently it should be scanned. By delaying the scanning of low risk files in this way, the antivirus suffers just a slight decrease in the security level it provides. Conversely, by increasing the scanning frequency for high risk files, the security level increases. Overall, this optimization strategy can fine-tune (and even enhance) the security level of antivirus scans while maintaining a manageable energy toll.

We propose estimating the risk of a file by: (i) relying on its meta-data; (ii) analyzing the user's behavior; and (iii) leveraging threat intelligence.

The meta-data attributes of the file to take into account are the file's type, permissions, owner, version, and source (e.g., the ADS (alternate data stream) attribute of NTFS file system, which indicates whether the file was downloaded from the web).

We can also predict (based on the user's behavior) when the user will use the file in the future. This way, we can avoid scanning this file in cases in which the file is predicted to be left untouched in the foreseeable future.

Threat intelligence can also be used to adjust the risk profile of a file dynamically (for instance, in cases of a new campaign that targets similar organizations with a vector that utilizes macros in Excel files); then the scan intervals can be adjusted accordingly. When scheduling scan intervals for low risk files, the antivirus may consider the optimizations proposed in Section 6.3. It should be noted that the energy cost associated with the implementation of the risk-based scan frequency optimization strategy should also be considered. For instance, in cases in which an organization plans to deploy a machine learning model that classifies the risk profile of a file in order to determine its required scan frequency, there is a need to calculate the energy consumption associated with training and running the model vis-à-vis the expected energy saving during its years of service. Due to the variety of model implementations and antiviruses, such calculations should be performed by the organization, taking into account the specific architectural details, and verified empirically.

6.2.2. Signature Scan Repetition Avoidance Across Time. This optimization strategy aims at reducing the energy toll stemming from redundant antivirus scans across time. As mentioned in Section 2.2, redundant antivirus scans occur in cases in which the file being scanned has not changed since its previous scan. In such cases, it is not reasonable to assume that the same input file would be classified differently by the antivirus at a later time, unless new signatures relevant to the file were published by the antivirus vendor. Therefore, prior to scanning a file we suggest checking whether the file has changed since its last scan or new signatures relevant to the file were added. If either has occurred, the file should be scanned; if neither occurred, it would be unreasonable to scan the file again using the same signatures and expect a different result. If the file was changed, it should be scanned again, using all relevant signatures. In cases in which new signatures were added but the file remained unchanged, the file should only be scanned against the new signatures, thereby reducing the number of signatures used, the required scan time, and the energy consumption.

File modification detection: One way of checking whether the file has changed since the last scan is by observing the timestamps of the file's meta-data; for example, the "modified" timestamp of NTFS stores information about the last time the file was modified; a timestamp that is later than the last scan indicates that the file has been modified after the last scan. However, relying solely on these timestamps to determine whether or not to run a scan can be dangerous, as an attacker can modify the timestamps found in the file's meta-data by performing a timestomping attack to manipulate the antivirus. Some of the tools used to execute such attacks were analyzed in [20].

Timestomping attacks: Since our proposed antivirus optimization relies on timestamps, an attacker could potentially modify these timestamps in such a way that it would appear that the file's content was last modified a long time ago in order to evade another antivirus scan (unless new relevant signatures were added). In Appendix D.2, we highlight several common methods for dealing with this problem in modern file systems and discuss their advantages and disadvantages in terms of security and energy consumption. By incorporating timestomping detection into our proposed optimization strategies, we reduce the risk of scan evasion but increase the energy toll of the solution.

6.2.3. Selective Signature Scan. This optimization component is responsible for the intelligent selection of signatures against which the file will be scanned. For instance, there is no need to scan a JPG file with a signature of malware that only affects executable files. For each signature in the antivirus database, we propose adding meta-data about the malware's target file type. This way, only the relevant signatures will be selected for each file before scanning it, rather than automatically scanning against all the signatures. It should be noted that some antiviruses already implement this approach. By combining the use of this component and the previous optimization component, we can significantly reduce the number of signatures required when scanning each file. For example, in cases in which a file has not changed since the last scan and the antivirus vendor has not published any new signatures that are relevant to the file, the antivirus can ignore this file.

To summarize the point of the proposed antivirus-specific optimizations, we propose a three-fold approach. First, we suggest determining the required scanning frequency based on the risk identified for a specific file (or a class of files). Second, by using only the new signatures in cases in which no changes were detected in the file, we avoid redundant scanning. Third, we further reduce the number of signatures by selecting the relevant signatures pertaining to the file from the remaining signatures.

6.3. Schedule Cybersecurity Scanning During Working Hours

Based on empirical experiments performed as a part of our study, we observed that when the computational load on the computer is low, running cybersecurity processes alongside other processes is more efficient in terms of energy consumption than running them at other times. Therefore, we propose running cybersecurity-related tasks in parallel with other daily tasks. The rational for this approach is rooted in the fact that oftentimes antivirus and other security software (e.g., backups) are run overnight in order to avoid disturbing computer users. Running these tools after hours requires waking the machine up from sleep mode, which consumes more energy than running the task in parallel with other jobs on an active machine. It increases the total amount of time the machine is active, which contributes to the overall energy toll. The energy savings that can be achieved by running antiviruses during working hours is further discussed in Appendix D.3. It should be noted that this recommendation applies to organizations that pay a constant rate per kWh. In some regions, electric companies charge a higher rate for daytime electricity use and a reduced rate for use at night; in such cases, an empirical evaluation should be performed to determine the optimal cost optimization strategy. Such rate variations can also be leveraged in the online/offline optimization presented in Section 3.2. For example, as many computations (pre-calculations) as possible could be transferred to the offline phase, which could be scheduled when the energy rate is low.

7. Antivirus Optimization Evaluation

In this section, we evaluate our proposed antivirus optimizations, employing several popular antiviruses as our test cases. Specifically, we used ClamAV [71], which is a popular cross-platform open-source antivirus (this was chosen, since we can view its signature database and modify its source code to implement the optimizations proposed in Section 6.2); Sophos antivirus, which is a popular antivirus with a market share estimated at 19% [2]; and Microsoft Defender, which is a built-in antivirus suite for Microsoft Windows systems.

We performed our evaluation of the first optimization strategy presented in Section 6.2.1, risk-based scan frequency, with all three antivirus test cases (ClamAV, Sophos, and Microsoft Defender). Our evaluation of the second optimization strategy presented in Section 6.2.2, signature scan repetition avoidance across time, was only performed with ClamAV, since it requires modification of the internal logic of the antivirus, which we accomplished by modifying the source code. The evaluation of the third optimization strategy presented in Section 6.2.3, selective signature scan, could not be performed with the Sophos and Microsoft Defender antivirus test cases because of the need for access to the internal logic. When ClamAV was used in our evaluation, we observed that it already partially implements this optimization component, and therefore we did not evaluate the impact of such optimization in full. ClamAV implements this component by saving the target file type in its signature database. When a file is scanned, signatures that do not match the file type are ignored.

TABLE 3. SCAN TIME AND ENERGY SAVINGS RESULTING FROM THE RISK-BASED SCAN FREQUENCY OPTIMIZATION STRATEGY. LOW, MEDIUM, AND HIGH RISK FILES ARE SCANNED EVERY 14, 10, AND 6 DAYS RESPECTIVELY. PRIOR TO OPTIMIZATION, ALL FILES ARE ASSUMED TO HAVE BEEN SCANNED WEEKLY.

Antivirus	Scan time savings	Energy consumption savings
ClamAV	6.44%	5.51%
Sophos	6.42%	6.40%
Microsoft Defender	12.19%	13.20%

7.1. Risk-Based Scan Frequency Evaluation

We evaluated the first antivirus optimization strategy by dividing the files into three categories of risk based on their file type: (i) low risk files, such as data container files, which include CSV, TXT, and images; (ii) medium risk files, which include data container files with code comingling capability, such as DOC, PPT, PDF, and HTML; and (iii) high risk files, which contain code, such as DLL and EXE. As explained in detail in Appendix D.1, we calculated the time and energy required to scan 1 MB of files from each risk category by each of the examined antivirus test cases. Then, we factored our results by the size proportion of each risk category, under the assumption that the total size of each risk category is equivalent to that of the other categories. We also assume that a regular engine scans all of the files once a week, while an optimized engine scans the low, medium, and high risk files based on a different frequency, such as once every 14, 10, and 6 days respectively. Based on those assumptions, we calculated the antiviruses' scan time and total energy consumption for a month-long period (30 days). The total savings (scan time and energy consumption) resulting from this optimization strategy are summarized in Table 3.

It should be noted that our assumptions are not fixed; they can be tailored to the circumstances of each organization. As can be seen in Appendix D.1, some file types are "costly" to scan in terms of their energy requirements per MB. Therefore, to achieve the optimal tradeoff between security and energy consumption, we suggest that each organization consider its security policy, along with each file type's share of the overall storage size within the organization, when selecting a suitable antivirus solution.

7.2. Signature Scan Repetition Avoidance Across Time Evaluation

To evaluate the signature scan repetition avoidance strategy, we obtained two different snapshots of the same signature database by using ClamAV's native *freshclam* program. The first snapshot, denoted as $DB_{old-ver}$, represents the signature database used in the earlier antivirus scan. The second snapshot, which was taken a week later and is denoted as $DB_{new-ver}$, represents an updated signature database containing additional signatures published by ClamAV. Next, we created a "delta" database using the *sigtool* program. This database, denoted as DB_{diff},

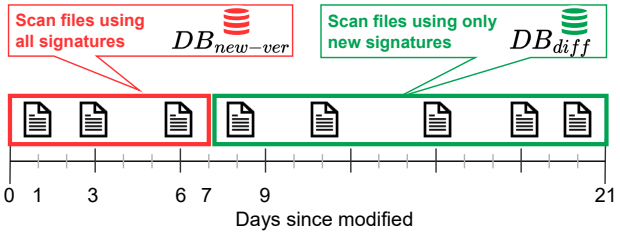

Figure 4. Illustration of temporal scan repetition avoidance optimization evaluation. We assume that an antivirus scan was performed seven days ago.

consists only of the signatures that were added or changed from $DB_{old-ver}$ to $DB_{new-ver}$, meaning that these signatures were not available in the previous scan.

To implement this optimization strategy, when performing a new antivirus scan, only $DB_{new-ver}$ and DB_{diff} are taken into account. Namely, upon scanning a file that has been modified since the previous scan, the entire $DB_{new-ver}$ is applied against the file. Otherwise (i.e., no file modification was detected), only signatures in DB_{diff} are used to scan the file. Our implementation method is presented in Figure 4 and available as open-source [1]. In order to determine when the file was last modified, the "modified" timestamp from the $STANDARD_INFORMATION ($SI) attribute is read. However, as noted in Section 6.2.2, relying on just this timestamp enables an attacker to perform a timestomping attack. Therefore, we implemented an optional basic timestomping detection countermeasure as a defensive strategy. Specifically, since most timestomping tools allow alterations of file timestamps with limited precision, we tagged files as suspicious in cases in which their "modified" timestamp contains a trailing sequence of zeros. These suspicious files underwent a comprehensive scan using the signatures in $DB_{new-ver}$, regardless of what their "modified" timestamp says. Appendix D.2 provides a more comprehensive description of timestomping, including detection techniques.

Table 4 presents the scan time and energy consumption when a ClamAV scan was performed on an entire C drive in three different scenarios: (i) before applying the optimization strategy; (ii) after applying the optimization strategy without timestomping detection; and (iii) after applying the optimization strategy with timestomping detection. As seen in the table, the use of this optimization strategy dramatically reduces the energy consumption and running time of the antivirus. Compared to the original implementation of ClamAV, the use of the optimization strategy without timestomping detection reduces the run time and energy consumption by 75.14% and 76.61% respectively, while the use of the optimization strategy with timestomping detection reduces the running time and energy consumption by 47.51% and 47.84% respectively.

The results of our evaluation show that the proposed optimization strategies can reduce the amount of energy consumed by antiviruses.

TABLE 4. EVALUATION RESULTS: SIGNATURE SCAN REPETITION AVOIDANCE OPTIMIZATION STRATEGY. CLAMAV SCANNED THE ENTIRE C DRIVE OF A TYPICAL WINDOWS-BASED LAPTOP. AS AN OPTIONAL BASIC TIMESTOMPING COUNTERMEASURE, WE ANALYZED THE $SI MODIFICATION TIMESTAMP PRECISION.

Version	Scan time (seconds)	Energy consumption (mWh)
Original ClamAV	86,532.56	739,310
Optimized ClamAV	21,508.68	172,900
Optimized ClamAV with Timestomping Detection	45,420.89	385,640

8. Conclusion

In this paper, we highlighted the need for energy-aware cybersecurity countermeasures and shed light on scenarios where cybersecurity may be inefficient. We presented a Green Security taxonomy based on a review of methods used to measure and optimize the energy consumption of cybersecurity solutions. We also introduced a novel and robust measurement framework for estimating the energy consumption of each process, where processes are treated as black-boxes. This framework is composed of a machine learning model capable of predicting the energy consumption of a process based on its resource consumption (e.g., CPU, RAM, disk use), while taking the hardware and operating system specifications into account. With this model, an organization can assess the energy consumption of various processes, including cybersecurity solutions. Measurement of its cybersecurity energy consumption opens the door to discussion regarding potential optimizations and future energy savings. We also proposed security-related energy optimizations. For antivirus solutions, we laid out the following principles:

1) Not every signature is suitable for every file type; therefore, it is important to ensure that each file is only scanned against the relevant signatures.
2) Scanning a file with the same signatures produces the same result; therefore, only new relevant signatures need to be used for scanning if the file has not been modified since the last scan; in this regard, we pointed out the potential risks (e.g., timestamping) and proposed countermeasures.
3) Antiviruses can reduce the frequency of scanning low risk files to save energy while maintaining the required risk profile.

We pointed out (and illustrated in Appendix D.3) that organizations can save energy and money by rescheduling cybersecurity scans (such as routine antivirus scans) during working hours instead of keeping computers awake solely for security tasks.

In future work, we plan to extend our research on optimizing the energy consumption of security solutions to additional security tools and expand our taxonomy so that it covers other environments (such as the cloud environment). We also aim to enhance our measurement framework by expanding our dataset with more samples. Another future extension relates to explainable artificial intelligence (XAI) [9], which could be applied to our measurement model. More specifically, XAI can be used to better understand how CPU usage, memory allocations, and disk operations affect the energy consumption of a running process.

References

[1] Measurement program. https://github.com/sapirani/GreenSecurityMeasurementAndOptimizationFramework.

[2] 6sense. Market share of sophos. https://6sense.com/tech/antivirus/sophos-market-share.

[3] Faitouri A Aboaoja, Anazida Zainal, Fuad A Ghaleb, Bander Ali Saleh Al-rimy, Taiseer Abdalla Elfadil Eisa, and Asma Abbas Hassan Elnour. Malware detection issues, challenges, and future directions: A survey. *Applied Sciences*, 12(17):8482, 2022.

[4] Kazi Main Uddin Ahmed, Math H. J. Bollen, and Manuel Alvarez. A review of data centers energy consumption and reliability modeling. *IEEE Access*, 9:152536–152563, 2021.

[5] Mohammed Al-Asli and Taher Ahmed Ghaleb. Review of signature-based techniques in antivirus products. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–6. IEEE, 2019.

[6] Jennifer Allen. 5 reasons you should run an antivirus scan at least once each week. https://www.windowscentral.com/5-reasons-you-should-run-antivirus-scan-least-once-each-week, 2020.

[7] Tibra Alsmadi and Nour Alqudah. A survey on malware detection techniques. In *2021 International Conference on Information Technology (ICIT)*, pages 371–376. IEEE, 2021.

[8] Anders S. G. Andrae and Tomas Edler. On global electricity usage of communication technology: Trends to 2030. *Challenges*, 6(1):117–157, 2015.

[9] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[10] Ömer Aslan Aslan and Refik Samet. A comprehensive review on malware detection approaches. *IEEE access*, 8:6249–6271, 2020.

[11] Avira. How often should i scan my pc for malware? https://www.avira.com/en/blog/how-often-should-i-scan-my-pc-for-malware, 2019.

[12] Zahra Bazrafshan, Hashem Hashemi, Seyed Mehdi Hazrati Fard, and Ali Hamzeh. A survey on heuristic malware detection techniques. In *The 5th Conference on Information and Knowledge Technology*, pages 113–120. IEEE, 2013.

[13] Jeffrey Bickford, H Andrés Lagar-Cavilla, Alexander Varshavsky, Vinod Ganapathy, and Liviu Iftode. Security versus energy tradeoffs in host-based mobile malware detection. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 225–238, 2011.

[14] Yoni Birman, Shaked Hindi, Gilad Katz, and Asaf Shabtai. Cost-effective ensemble models selection using deep reinforcement learning. *Information Fusion*, 77:133–148, 2022.

[15] Luca Caviglione, Mauro Gaggero, Enrico Cambiaso, and Maurizio Aiello. Measuring the energy consumption of cyber security. *IEEE Communications Magazine*, 55(7):58–63, 2017.

[16] Luca Caviglione, Alessio Merlo, and Mauro Migliardi. What is green security? In *2011 7th international conference on information assurance and security (IAS)*, pages 366–371. IEEE, 2011.

[17] Ben Cutler, Spencer Fowers, Jeffrey Kramer, and Eric Peterson. Dunking the data center. *IEEE Spectrum*, 54(3):26–31, 2017.

[18] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):732–794, 2016.

[19] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S Blair, and Adrian Friday. The real climate and transformative impact of ict: A critique of estimates, trends, and regulations. *Patterns*, 2(9):100340, 2021.

[20] Michael Galhuber and Robert Luh. Time for truth: Forensic analysis of ntfs timestamps. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–10, 2021.

[21] Erol Gelenbe. Energy consumption by ict and cybersecurity at the time of cop26. https://iotac.eu/energy-consumption-by-ict-and-cybersecurity-at-the-time-of-cop26/, 2021.

[22] Tim Golden. Wmi. https://pypi.org/project/WMI/.

[23] Dries R. Goossens and Frederik Spieksma. Exact algorithms for procurement problems under a total quantity discount structure. *Eur. J. Oper. Res.*, 178:603–626, 2007.

[24] Reinhard Grünwald and Claudio Caviezel. Energy consumption of ict infrastructure. Technical report, Büro für Technikfolgen-Abschätzung beim Deutschen Bundestag (TAB), 2022. 46.24.01; LK 01.

[25] Chen Guo, Yang Yang, Yanglin Zhou, Kuan Zhang, and Song Ci. A quantitative study of energy consumption for embedded security. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–5. IEEE, 2021.

[26] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. Measuring energy consumption for short code paths using rapl. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):13–17, 2012.

[27] Basel Halak, Yildiran Yilmaz, and Daniel Shiu. Comparative analysis of energy costs of asymmetric vs symmetric encryption-based security applications. *IEEE Access*, 10:76707–76719, 2022.

[28] Ather Hassan, Syed Zafar Ilyas, Abdul Jalil, and Zahid Ullah. Monetization of the environmental damage caused by fossil fuels. *Environmental Science and Pollution Research*, 28:21204–21211, 2021.

[29] Hamed Hellaoui, Mouloud Koudil, and Abdelmadjid Bouabdallah. Energy-efficient mechanisms in security of the internet of things: A survey. *Computer Networks*, 127:173–189, 2017.

[30] INRIA. scikit-learn. https://scikit-learn.org/stable/.

[31] SANS Institute. Kansa for enterprise-scale threat hunting. https://www.sans.org/presentations/kansa-for-enterprise-scale-threat-hunting/, 2023.

[32] Kaspersky. How to run a virus scan the right way: Step-by-step guide. https://usa.kaspersky.com/resource-center/preemptive-safety/how-to-run-a-virus-scan.

[33] Avita Katal, Susheela Dahiya, and Tanupriya Choudhury. Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing*, pages 1–31, 2022.

[34] R. Kauffman and J. Y. Tsai. The unified procurement strategy for enterprise software: A test of the "move to the middle" hypothesis. *Journal of Management Information Systems*, 26:177 – 204, 2009.

[35] Phongsak Keeratiwintakorn and Prashant Krishnamurthy. Energy efficient security services for limited wireless devices. In *2006 1st International Symposium on Wireless Pervasive Computing*, pages 1–6. IEEE, 2006.

[36] Keysight. How to select the right current probe. https://www.keysight.com/us/en/assets/7018-05961/application-notes/5992-2656.pdf?success=true, 2018.

[37] Nam Sung Kim, Todd Austin, David Baauw, Trevor Mudge, Krisztián Flautner, Jie S Hu, Mary Jane Irwin, Mahmut Kandemir, and Vijaykrishnan Narayanan. Leakage current: Moore's law meets static power. *computer*, 36(12):68–75, 2003.

[38] Karthik Kumar and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56, 2010.

[39] Jongdeog Lee, Krasimira Kapitanova, and Sang H Son. The price of security in wireless sensor networks. *Computer Networks*, 54(17):2967–2978, 2010.

[40] Xun Li and Frederic T Chong. A case for energy-aware security mechanisms. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 1541–1546. IEEE, 2013.

[41] Steve Mansfield-Devine. Do you have too much security? *Network Security*, 2023(3), 2023.

[42] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020.

[43] Katrina Matthews. Anti-virus software: How often to run and why? https://www.colocationamerica.com/blog/anti-virus-software-how-often-to-run-and-why, 2014.

[44] ICF McAfee. The carbon footprint of email spam report, 2009.

[45] Alessio Merlo, Mauro Migliardi, and Luca Caviglione. A survey on energy-aware security mechanisms. *Pervasive and Mobile Computing*, 24:77–90, 2015.

[46] Microsoft. Use group policy to install software. https://learn.microsoft.com/en-us/troubleshoot/windows-server/group-policy/use-group-policy-to-install-software, 2023.

[47] Wicher Minnaard, CTAM de Laat, and M van Loosen MSc. Timestomping ntfs. *IMSc final research project report, University of Amsterdam, Faculty of Natural Sciences, Mathematics and Computer Science*, 2014.

[48] Sparsh Mittal and Jeffrey S. Vetter. A survey of methods for analyzing and improving gpu energy efficiency. *ACM Comput. Surv.*, 47(2), aug 2014.

[49] Alji Mohamed and Chougdali Khalid. Detection of timestamps tampering in ntfs using machine learning. *Procedia Computer Science*, 160:778–784, 2019.

[50] U.S. Bureau of Labor Statistics. Electricity per kwh in u.s. https://data.bls.gov/timeseries/APU000072610?amp%253bdata_tool=XGtable&output_view=data&include_graphs=true, 2023. Accessed: 2023-05-30.

[51] David Palmbach and Frank Breitinger. Artifacts for detecting timestamp manipulation in ntfs on windows and their reliability. *Forensic Science International: Digital Investigation*, 32:300920, 2020.

[52] UK Parliament. Energy consumption of ict. https://researchbriefings.files.parliament.uk/documents/POST-PN-0677/POST-PN-0677.pdf, 2022.

[53] Sylvain Pelissier, Tamma V Prabhakar, Hirisave Shankaralingaiah Jamadagni, R VenkateshaPrasad, and Ignas Niemegeers. Providing security in energy harvesting sensor networks. In *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 452–456. IEEE, 2011.

[54] Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. Ranking programming languages by energy efficiency. *Science of Computer Programming*, 205:102609, 2021.

[55] Frederica Perera and Kari Nadeau. Climate change, fossil-fuel pollution, and children's health. *New England Journal of Medicine*, 386(24):2303–2314, 2022.

[56] Iasonas Polakis, Michalis Diamantaris, Thanasis Petsas, Federico Maggi, and Sotiris Ioannidis. Powerslave: Analyzing the energy consumption of mobile antivirus software. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 165–184. Springer, 2015.

[57] Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on mobile computing*, 5(2):128–143, 2005.

[58] Phongsak Prasithsangaree and Prashant Krishnamurthy. On a framework for energy-efficient security protocols in wireless networks. *Computer Communications*, 27(17):1716–1729, 2004.

[59] Murad A Rassam, Mohd Maarof, Anazida Zainal, et al. Big data analytics adoption for cybersecurity: A review of current solutions, requirements, challenges and trends. *Journal of Information Assurance & Security*, 12(4), 2017.

[60] Giampaolo Rodola. Psutil. https://github.com/giampaolo/psutil.

[61] Sanjay K Sahay, Ashu Sharma, and Hemant Rathore. Evolution of malware and its detection techniques. In *Information and Communication Technology for Sustainable Development: Proceedings of ICT4SD 2018*, pages 139–150. Springer, 2020.

[62] Yosra Ben Saied, Alexis Olivereau, Djamal Zeghlache, and Maryline Laurent. Lightweight collaborative key establishment scheme for the internet of things. *Computer Networks*, 64:273–295, 2014.

[63] Mattia Salnitri, Pierluigi Plebani, and Alessandra Raffone. Towards designing energy-aware cybersecurity policies. In *International Conference on Advanced Information Systems Engineering*, pages 26–33. Springer, 2023.

[64] Kaushik Chandra Deva Sarma and Santanu Sharma. A method for reduction of off state leakage current in symmetric dg jlt. *Engineering Research Express*, 1(1):015034, 2019.

[65] Daniela N Schmidt. *Summary for Policymakers: Climate change 2022: Impacts, adaptation and vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel of Climate Change (IPCC),*. Cambridge University Press, United Kingdom, February 2022.

[66] Vijay Kumar Sharma. A survey of leakage reduction techniques in cmos digital circuits for nanoscale regime. *Australian journal of electrical and electronics Engineering*, 18(4):217–236, 2021.

[67] Jagsir Singh and Jaswinder Singh. A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture*, 112:101861, 2021.

[68] Daniel J Soeder and Daniel J Soeder. Fossil fuels and climate change. *Fracking and the Environment: A scientific assessment of the environmental risks from hydraulic fracturing and fossil fuels*, pages 155–185, 2021.

[69] JAM software. Heavyload. https://www.jam-software.com/heavyload.

[70] Antonio Vincenzo Taddeo, Marcello Mura, and Alberto Ferrante. Qos and security in energy-harvesting wireless sensor networks. In *2010 International conference on security and cryptography (SECRYPT)*, pages 1–10. IEEE, 2010.

[71] Cisco Talos. Clamav. https://www.clamav.net/.

[72] Ian Tiseo. Annual carbon dioxide (co_2) emissions worldwide from 1940 to 2023. https://www.statista.com/statistics/276629/global-co2-emissions/, 2023.

[73] Kevin Townsend. Are overlapping security tools adversely impacting your security posture? https://www.securityweek.com/are-overlapping-security-tools-adversely-impacting-your-security-posture/, 2020.

[74] Evangelos Vasilakis. An instruction level energy characterization of arm processors. *Foundation of Research and Technology Hellas, Inst. of Computer Science, Tech. Rep. FORTH-ICS/TR-450*, 2015.

[75] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, and Hojung Cha. {AppScope}: Application energy metering framework for android smartphone using kernel activity monitoring. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 387–400, 2012.

[76] Patrycjusz Zdzichowski, Michal Sadlon, Teemu Uolevi Väisänen, Alvaro Botas Munoz, and Karina Filipczak. Anti-forensic study. *NATO CCD COE, Tallinn*, 2015.

[77] David Zeuthen. Wmi. https://manpages.ubuntu.com/manpages/jammy/man1/upower.1.html.

[78] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. Electricity bill capping for cloud-scale data centers that impact the power markets. In *2012 41st International Conference on Parallel Processing*, pages 440–449. IEEE, 2012.

Appendices

A. Background

In this appendix, we provide a brief overview of how traditional antivirus software mitigates malicious activities. In section 6 and section 7, we measure the energy consumption of several antivirus controls, propose means of optimizing them, and evaluate the effectiveness of the optimizations.

Antivirus is a common cybersecurity control that aims to detect and remove malicious software (referred to as malware), such as viruses, spyware, adware, Trojanized software, and computer worms. In order to detect malware, modern antiviruses use a combination of signature-, heuristic-, and behavioral-based techniques [3], [7], [12].

In signature-based detection, antivirus vendors maintain a database of signatures, where each signature represents a malware fingerprint. Such fingerprints are usually comprised of a sequence(s) of bits that the antivirus agent looks for at a specific offset(s) of the file [5]. When an antivirus scans a file, it searches for these fingerprints in the file. Upon identifying a match, the antivirus issues an alert indicating that it has found malware. However, signature-based approaches are unable to detect new malware, since the fingerprints are not yet available. The bulk of modern malware is polymorphic so as to avoid signature-based detection. In such cases, each copy of the malware is somewhat different from other copies from a raw binary perspective, although all copies of the malware preserve the same semantics and core functionality. Since the binary differs between different instances of the same malware, signature-based detection techniques are quite limited.

To overcome this limitation, modern antiviruses employ heuristic- and behavioral-based techniques which do not require knowledge of the malware's fingerprint [3], [10]. In a heuristic-based approach, the antivirus uses a database of predefined generic rules. Each rule defines a score for a suspicious action (e.g., deleting or encrypting many files, connecting to external addresses, and sending information out). When analyzing a certain file (whether by static or dynamic analysis), each suspicious action performed by that file is given a score, which is aggregated across all suspicious actions of the file. Upon passing a certain threshold, the file under test is flagged as malicious [61].

In the behavioral-based approach, the file under test runs in a controlled and isolated environment such as a sandbox, where the antivirus tracks the operations it performs (e.g., the file modifications it attempts to make, the processes it interacts with) After extracting features from the tracked operations, machine learning methods (such as anomaly detection) can be applied to determine whether the file under test is benign or malicious [67].

While the signature-based approach is very accurate in flagging malicious files with a low false positive rate, for obvious reasons the heuristic- and behavioral-based approaches have a higher false positive rate.

An antivirus scan can be run using various scan modes. For most antiviruses, the user can select the scan mode. The most common scan modes are a "full scan," where all folders and files are scanned; a "quick scan," where the antivirus vendor's predefined subset of folders and files is scanned; and a "custom scan," where the user's predefined subset of folders and files is scanned. Obviously, the more files and folders scanned, the greater the energy toll.

Antivirus vendors recommend performing scans regularly, but most do not specify what "regularly" means. Based on our review of various online forums on cybersecurity as well as blog posts and community discussions, it seems that a typical recommendation is to run a quick scan on a daily basis, in addition to a weekly full scan [6], [11], [32], [43].

B. Extended Related Work

In this appendix, we expand our discussion on the efforts being made to measure and optimize the energy consumption of cybersecurity. We also discuss earlier research that highlights the energy/security tradeoffs, enabling the reduction of energy consumption while taking a conscious risk.

B.1. Energy Consumption Measurement

In this subsection, we provided a detailed review of prior research on the measurement of cybersecurity tools' energy consumption.

B.1.1. Cryptography. Interestingly, within the context of cybersecurity energy efficiency, a significant body of work focused on measuring the energy consumption of cryptography algorithms, including the energy analysis of different encryption algorithms (RSA, DSA, AES, ECDSA, etc.) and hash functions (SHA1, SHA256, MD5, etc.) [15], [25], [27], [39], [40], [57], [58].

In this respect, [29] came to the conclusion that heavy computations (such as exponent computations in the Diffie-Helman key exchange algorithm), the amount of data that the cryptographic mechanism operates on, and the number of invocations of the cryptographic mechanism (e.g., the number of times a key is created) are the most significant factors when determining the power consumption related to a cryptographical mechanism.

Similarly, a large number of papers analyzed the energy overhead of transport security [27], [57]. One way of minimizing the energy consumed when transporting data through an encrypted channel is to change the size of the packets sent, as discussed in [57], since the handshake protocol, a part of the TLS protocol, can be optimized when sending short messages by optimizing asymmetric encryption algorithms. The results of this study also showed that using an RSA-based SSL handshake is much more efficient than an ECC-based handshake when there is no client authentication.

In [27], the authors estimated the global energy costs of TLS for web browsing based on the Internet's energy consumption and the amount of data used per year for both symmetric and asymmetric algorithms. The results align with other studies that determined that asymmetric encryption is more energy wasteful than symmetric encryption in most typical settings.

Another approach for estimating the energy consumption of popular cryptography algorithms was presented in [25]. The authors monitored the assembly instructions

invoked by three different security programs (which implemented different encryption algorithms) during runtime and counted the number of appearances of each such assembly instruction. Then, they multiplied the energy consumption of each assembly instruction [74] by the number of appearances of the instruction during runtime and totaled the results.

Although this approach differentiates the energy consumed by cybersecurity software from other device operations, it does not take into account cache misses (e.g., situations where the CPU does not have the relevant data available in the cache and has to fetch it from the main memory to execute the assembly instruction, a process in which more energy is expended) and other optimizations which may affect the energy consumption. In addition, the authors had access to the source code and compiled it with a debugging flag on, so they could debug it at the instruction level.

B.1.2. Energy Consumption of Specific Cybersecurity Mechanisms. Other researchers have explored the energy consumption of specific cybersecurity solutions. One paper worth mentioning is [16] in which the authors made an initial effort to evaluate the energy consumption of IDSs and coined the term green security. They evaluated the energy consumption of two different strategies for sending packets between two routers while analyzing them using an IDS. In the first strategy, referred to as "the centralized strategy" by the authors, packets are transferred directly from the source router to the destination router which includes an IDS capability that analyzes the incoming packets. In the second strategy, which is called "the closed-loop strategy," the data is first transferred through a closed loop of routes. Each router in the loop has an IDS capability that performs a portion of the overall analysis; when the packet reaches the end of the loop, the analysis is completed. Packets deemed benign are routed out of the loop to the destination router, while the packets flagged as malicious by any IDS in the loop are dropped from the loop and not forwarded on. According to the authors, the energy efficiency of the closed-loop strategy increases when the amount of malicious packets increases, since there is no energy toll resulting from routing these packets, as opposed to the centralized strategy, whose energy efficiency does not increase in those circumstances. The size of the closed loop and the distance to the destination also play a role in calculating the pros and cons of each approach in terms of the energy toll.

In a broader context, [40] estimated the energy required to process one byte of data by several cybersecurity mechanisms (e.g., spam filtering, antivirus). The authors used various industry benchmarks to derive their results. For instance, to calculate the energy consumed (per byte) by a McAfee spam filter, they relied on a McAfee report that provides the annual energy consumption of a spam filter [44] and divided that value by the total amount of traffic (in bytes) that passed through the spam filter annually. Using these estimations, the paper concluded that the energy used by cybersecurity solutions represents almost 30% of the energy demand associated with computations.

To complete the picture, [63] aimed to estimate the total energy consumption of cybersecurity mechanisms in an organization. The authors proposed a method for modeling the organizational security policy that determines the security objectives and technical cybersecurity controls deployed to enforce the policy. They suggested relying on existing methods to estimate the energy consumption of each cybersecurity mechanism and totaled them to assess the total energy consumed by organizational cybersecurity solutions. By using this kind of modeling, an organization should be able to begin to assess the total energy consumption of its cybersecurity controls and plan the security policy accordingly. Since this method relies on other methods to measure the actual energy consumption of the security solutions, it can only serve as as a mental model rather than a self-sufficient energy measurement tool.

B.2. Non-Redistributive Energy Consumption Optimization

The organizational context may play a role in devising optimization methods. Namely, in some cases, we should consider the specific architecture, type of network traffic, and other parameters that vary across different organizations. The papers below present opportunities for optimizing energy consumption by tailoring the enterprise's security countermeasures to its specific characteristics.

In [16], two distributed intrusion detection system (DIDS) policies were examined, and the authors found that the energy consumption related to each policy depended on the network architecture and traffic characteristics.

Another study that found that security solutions can be identical from a security level perspective but differ in terms of their energy consumption is [58]; the authors demonstrated that different packet sizes affect the energy consumption of various cryptography algorithms differently.

Such variance implies that in some cases there is a need to adapt the optimization techniques (and the underlying security controls) to the idiosyncrasies of the organization (i.e., 'adaptive security') [29].

B.3. Energy/Security Tradeoff

Unsurprisingly, the energy/security tradeoff is particularly prominent in the field of cryptography. According to [58] and [57] and as discussed earlier, increasing the key size of various cryptography algorithms has a significant negative impact on an algorithm's energy consumption. The same is true for increasing the number of rounds of encryption. Therefore, instead of always resorting to the use of the strongest (and most energy consuming) encryption possible, [70] suggested classifying the sensitivity of the information and determining the necessary security level provided by the algorithm accordingly first. Put differently, the key size used by cryptography algorithms should be increased when protecting highly sensitive information and vice versa.

Modifying the security level according to the number of years that the information should be protected was suggested in [35]. More specifically, the authors proposed a scheme that adjusts the level of security (by reducing the computation in encryption algorithms) that satisfies specific security requirements and utilizes different security

algorithms based on the transmission packet size to further minimize the energy consumption of wireless devices.

Although we believe that the approach presented in both of the aforementioned papers is the the right path to follow, we acknowledge that it is a very narrow approach. Both studies aimed at reducing the energy consumption of security controls while maintaining a sufficient level of information security, and they did so by looking at a specific attribute (i.e., number of years to protect the data, sensitivity of the data) and tweaking a specific security control accordingly. However, to make this approach more general, prior to determining the configuration of the security control, there is a need to consider the entire threat model (e.g., the sensitivity of the data, as well as the intent and capabilities of the relevant adversary) and tune the energy/security tradeoffs accordingly.

In [14], the authors proposed a generic framework based on deep reinforcement learning (DRL) that learns how to minimize the detection cost (based on the processing time, monetary spending on cloud-based resources, etc.) by selecting the most appropriate machine learning detectors - out of a set of possible detectors - for a given situation, while maximizing the detection accuracy. This framework enables the end user to define the cost/accuracy tradeoff and prioritize cost reduction over accuracy or vice versa. The optimization process uses a state vector whose size is equal to the number of detectors. The i'th entry in the vector represents the classification confidence score of an analyzed sample according to the i'th detector. If a detector has not yet been applied, the relevant entry in the state vector is -1. In each iteration, the agent is able to select another detector to further analyze the sample or decide on the final classification after gaining sufficient confidence. In addition to demonstrating the proposed framework's ability to learn the user's preferences in terms of the cost/accuracy tradeoff under static conditions, the authors showed that their framework can easily adapt to changes, such as a degradation in the performance of some detectors. In the context of green security, the framework can be leveraged to optimize the energy consumption of organizational detection tools based on the organizational security policy.

C. Additional Energy Measurement Results

To validate our framework's ability to calculate the energy consumption of a process based on its resource consumption, we built several machine learning models and chose the best candidate, based on a comparison of their predictive performance.

First, we performed the experiments described in subsubsection 5.1.1. In each experiment, we executed a process that consumes a specific resource load combination, using the HeavyLoad [69] application as previously mentioned. This application allows us to determine the number of threads that the process will open, the priority of the process, the frequency of memory allocation, and the number of bytes written to a temporary file per second.

As mentioned in subsection 5.3, in order to evaluate the model's performance, we used the training set to train a regression model and a test set (80/20). We compared the performance of various regressors from the scikit-learn library [30], such as MLPRegressor, LinearRegressor, AdaBoostRegressor, and DecisionTreeRegressor, by performing cross-validation with grid search. Grid search was also used to fine-tune the hyperparameters of each regressor examined. As scoring functions, the grid search takes the negative mean absolute error (MAE) and the negative root mean squared error (RMSE) into account. The results of the grid search operation are presented in Table 5.

TABLE 5. THE PERFORMANCE OF THE EXAMINED REGRESSORS. THE MODELS WERE TRAINED USING A CROSS-VALIDATION TECHNIQUE. EACH ROW INCLUDES THE MAE AND RMSE VALUES OBTAINED BY THE MODEL DURING GRID SEARCH.

Model	RMSE	MAE
AdaBoostRegressor	182.59	141.90
ARDRegression	268.30	205.59
BayesianRidge	256.80	196.37
DecisionTreeRegressor	184.83	141.90
ElasticNet	252.05	194.09
ExtraTreeRegressor	187.39	144.36
GradientBoostingRegressor	182.44	142.19
HistGradientBoostingRegressor	181.21	140.63
RandomForestRegressor	176.89	134.06
PLSRegression	251.87	196.65

As can be seen in the table, the RandomForestRegressor obtained the best results in terms of the MAE and RMSE values. We performed additional experiments by training additional RandomForestRegressor models, using other parameter combinations and performing additional grid searches. The RandomForestRegressor models were trained with the "leave-one-out" training method, where in every iteration, the model is trained on the entire training set, except for one record that serves as a validation set.

Based on the results presented in Table 6, we determined that the best model for the energy prediction task is a RandomForestRegressor model, with the following parameters: max_depth=15, max_features='sqrt,' min_samples_split=5, and n_estimators=500, since this model achieved the lowest MAE and RMSE values (MAE = 139.664 and RMSE = 139.630). After identifying the model with the best performance on the training set (using grid search), the model was retrained on the entire training set; it was then used to predict the energy consumption of the processes included in the test set. The model achieved a MAE value of 134.50 and RMSE value of 166.88.

We also experimented with neural network models when examining models for our energy prediction task. We normalized the numeric features of the datasets and trained an MLPRegressor model with different parameters. The results were subpar compared to other models.

D. Optimization Results

In this appendix, we elaborate on our discussion on subsubsection 6.2.1, by estimating the time and energy required by various antiviruses to scan files from each risk category.

Then, we describe timestomping detection and mitigation techniques aimed at thwarting attackers that attempt to manipulate antiviruses by incorporating the optimization proposed in subsubsection 6.2.2.

Finally, we demonstrate the energy savings attainable by scheduling antivirus scans during active working hours,

TABLE 6. SUMMARY OF THE PERFORMANCE OF RANDOM FOREST REGRESSORS WITH DIFFERENT PARAMETERS ON THE TRAINING SET DURING THE GRID SEARCH OPERATION. EACH MODEL WAS TRAINED USING THE LEAVE-ONE-OUT TRAINING METHOD.

Parameters	RMSE	MAE
max_depth=15, max_features='sqrt', min_samples_split=5, n_estimators=500	139.630	139.664
max_depth=15, max_features='sqrt', min_samples_split=5, n_estimators=1000	140.329	140.480
max_depth=15, max_features='sqrt', min_samples_split=5, n_estimators=100	143.705	140.600
max_depth=10, max_features='sqrt', min_samples_split=5, n_estimators=500	140.784	141.049
max_depth=10, max_features='sqrt', min_samples_split=5, n_estimators=1000	140.654	141.334
max_depth=10, max_features='sqrt', min_samples_split=8, n_estimators=100	140.926	143.629

instead of running them overnight (as mentioned in subsection 6.3).

D.1. Estimation of the Time and Energy Required to Scan Each Risk Category

Here, we describe our method for estimating the time and energy required to scan 1 MB of each file risk category (low, medium, and high) by ClamAV, Sophos, and Microsoft Defender.

First, to ensure that our evaluation is representative of real-world environments, we selected several popular types of files to include in our evaluation and collected a dataset of benign files associated with those file types; then, we split the files into directories, each of which contains files associated with the same file type. For each directory, we measured the antivirus engines' energy consumption and scan time. Then, we divided the values obtained for each metric by the total size (in MB) of the respective directory, to account for the energy the antivirus invests per MB of scanned data. In other words, for each type of file, we calculated the amount of energy and time it took each antivirus engine to scan 1 MB of data. The results are presented in Table 7.

We estimated the energy required to scan 1 MB of low, medium, and high risk files, denoted by $E^{MB}_{low-risk}$, $E^{MB}_{medium-risk}$, and $E^{MB}_{high-risk}$. We also estimated the required scanning time per 1 MB of files from each risk category, denoted by $T^{MB}_{low-risk}$, $T^{MB}_{medium-risk}$, and $T^{MB}_{high-risk}$. For example, to estimate $E^{MB}_{low-risk}$, we first created a realistic profile of the total size of files for each type of low risk file, as observed on several typical laptop devices. Then, we extrapolated the results from Table 7 according to the distribution of the total size of each file type within the low risk category.

A similar process was conducted to estimate the other categories. The results are presented in Table 8.

D.2. Timestomping Detection and Mitigation Techniques

In this subsection, we provide an overview of timestomping detection and mitigation techniques aimed at impeding the attacker's ability to evade antivirus scans. In the NTFS file system, all timestamps have a precision point of 100-nanoseconds. However, most off-the-shelf timestomping tools can only modify timestamps at second-level granularity. Therefore, most timestomped timestamps will contain a trailing sequence of zeros, which is a strong indicator of timestomping, since the probability for such a timestamp to appear by chance is negligible. Another technique takes advantage of the $STANDARD_INFORMATION ($SI) and $FILE_NAME ($FN) attributes stored in NTFS master file table (MFT) records. Both $SI and $FN store the file's timestamps; $SI timestamps can be modified with the Windows API, but there is no API available to change the timestamps in $FN. In addition, when modifying the file's content, only $SI is updated, whereas $FN remains intact. Based on these differences, the detection technique identifies anomalies between the timestamps, such as in cases where $SI timestamps are older than $FN timestamps [76].

When combined, these techniques can detect the bulk of timestomping attacks [47], [49].

Another method for checking whether a file has changed without relying on timestamps is by fingerprinting the meta-data of all files the antivirus scanned (e.g., by hashing the MFT header and storing the result). In this technique, the antivirus calculates the fingerprint of the file's meta-data before scanning a file and compares the result to the latest recorded relevant fingerprint stored in its database. If they are identical, the antivirus assumes that the file has not been modified, since the meta-data should reflect any changes to the file. As in the case of timestomping, an attacker can tamper with the meta-data to evade antivirus scanning; however, reverting the meta-data to a previous state after changing the contents of a file may result in information loss and unpredictable behavior of the file, which may pose greater risk of being detected to the attacker. To take this further, the antivirus can maintain hashes of files' content instead of files' meta-data. This method increases the confidence that the file has not been modified, but it comes at the cost of higher energy consumption resulting from the need to process the entire file. Nevertheless, it is important to remember that the content hashes are calculated anyway by modern antiviruses for the purpose of checking hash signatures, and therefore this optimization can still reduce the overall energy consumed by an antivirus.

Another method an antivirus can use to identify files that were modified since the last scan leverages log files, such as $UsnJrl or $Logfile (which are maintained by Windows by default), to track any operation performed on any file, as discussed in [51]. To derive conclusions, log analytics techniques are used, making this the most energy-demanding alternative in the potential energy optimization arsenal. This point underscores the need to consider the specifics of the optimization techniques employed and measure the overall energy savings achieved in light of the techniques' energy requirements.

TABLE 7. TIME AND ENERGY REQUIRED BY CLAMAV, SOPHOS, AND MICROSOFT DEFENDER TO SCAN 1 MB OF VARIOUS **TYPES OF FILES**.

File type	ClamAV		Sophos		Microsoft Defender	
	Scan time (sec) per 1 MB	Energy (mWh) per 1 MB	Scan time (sec) per 1 MB	Energy (mWh) per 1 MB	Scan time (sec) per 1 MB	Energy (mWh) per 1 MB
csv	0.049708832	0.22587269	0.00672047	0.029774127	0.00212949	0.016735113
dll	0.78702162	6.168224299	0.092321049	0.451713396	0.010444311	0.080996885
doc	0.737310144	6.797752809	0.160961878	0.449438202	0.004583948	0.039606742
exe	2.589913226	23.63321799	0.427221853	2.006920415	0.072589853	0.460207612
html	0.22952621	0.76433121	0.028465654	0.184713376	0.008226866	0.071656051
images (jpg, png, gif)	0.385981641	3.39869281	0.032242434	0.163398693	0.006206037	0.051633987
manifest	17.01031192	99.03846154	4.554281533	17.30769231	1.574857274	9.134615385
pdf	0.962801993	9.511228534	0.099549347	0.607661823	0.004013455	0.04663144
ppt	0.668121019	5.994497937	0.038968882	0.220082531	0.003475141	0.025309491
txt	0.222313102	1.848739496	0.017765652	0.022408964	0.010645842	0.092156863
xls	0.767090308	6.847133758	0.131292462	0.828025478	0.013238716	0.073248408
xml	0.161282385	1.090047393	0.015024425	0.101895735	0.006307372	0.059004739

TABLE 8. TIME AND ENERGY REQUIRED BY CLAMAV, SOPHOS, AND MICROSOFT DEFENDER TO SCAN 1 MB OF EACH **RISK CATEGORY**.

Risk level	ClamAV		Sophos		Microsoft Defender	
	Scan time (sec) per 1 MB	Energy (mWh) per 1 MB	Scan time (sec) per 1 MB	Energy (mWh) per 1 MB	Scan time (sec) per 1 MB	Energy (mWh) per 1 MB
Low risk	0.33179769	2.15886496	0.05891853	0.256386891	0.021363897	0.145132836
Medium risk	0.747782351	6.938937816	0.090731493	0.474884053	0.005146219	0.050764977
High risk	1.387985489	11.98988886	0.203954651	0.970115736	0.031159492	0.207400461

TABLE 9. THE TOTAL ENERGY CONSUMED BY A TYPICAL LAPTOP COMPUTER SYSTEM (LENOVO YOGA I7) WHEN SIMULATING THE USE OF AN ANTIVIRUS (CLAMAV, SOPHOS, AND WINDOWS DEFENDER) OUTSIDE OF WORKING HOURS (I.E., THE USER ACTIVITY PROGRAM AND ANTIVIRUS RUN AT DIFFERENT TIMES) VERSUS DURING WORKING HOURS (I.E., THE USER ACTIVITY PROGRAM AND ANTIVIRUS RUN SIMULTANEOUSLY).

Running mode	ClamAV		Sophos		Microsoft Defender	
	Scan time (sec)	Energy (mWh)	Scan time (sec)	Energy (mWh)	Scan time (sec)	Energy (mWh)
Antivirus only	5943.755	54020	666.193	4790	100.386	670
User activity only	5944.051	43320	666.044	4050	100.040	540
Running at different times (antivirus runs outside of working hours)	11887.806	97340	1332.237	8840	200.426	1210
Running simultaneously (antivirus runs during working hours)	6694.312	55830	682.527	5030	107.299	890
Total savings	**43.69%**	**42.64%**	**48.77%**	**43.10%**	**46.46%**	**26.45%**

D.3. Schedule Cybersecurity Scans During Working Hours Evaluation

Here, we present the results obtained when optimizing the scheduling of cybersecurity processes (such as an antivirus scan), as proposed in subsection 6.3. We created a Python program called "user activity" that simulates the behavior of a typical user. The main activities performed by this program are creating MS Word, PowerPoint, and Excel files and inserting some content into the files, as well as searching the web and playing a YouTube video. The purpose of the program was to automate typical user activities, so that the user activity would be identical across our experiments, and the results would not be skewed by variance in human behavior. Next, we measured the energy consumed by the entire computer system when the user activity program was run. We also measured the energy consumed by the entire computer system while running just an antivirus scan (without running the user activity program). These two measurements simulate a case in which a standard user uses their computer during working hours, and when the user is no longer active (e.g., at night time), an antivirus scan is performed. Then, we measured the energy consumed by the entire computer system when running both the user activity program and an antivirus scan simultaneously. This measurement simulates a case where the antivirus performs a scan during working hours. We demonstrated our proposed scheduling optimization using ClamAV, Sophos, and Microsoft Defender. All of the antiviruses scanned the same dataset of files (described in subsection 7.1). The results are presented in Table 9 where it can be seen that for all of the antiviruses examined, the energy consumed when the user activity program and the antivirus scan were running at different times was significantly higher than when they were running simultaneously. This means that the energy consumption of a computer can be significantly reduced by running the antivirus scan during working hours. In addition, as seen in Table 8 and Table 9, there is a significant difference in the performance of different antiviruses. Specifically, Sophos and ClamAV have much longer scan times and higher energy consumption than Microsoft Defender. In fact, Sophos takes one order of magnitude longer for scanning and ClamAV takes two orders of magnitude longer for scanning, and their energy consumption is similarly higher.