Machine Learning Applications for Cybersecurity

M.A. Thomas

ABSTRACT

The trope of future cybersecurity as a battle between warring artificial intelligences awaits the development of artificial general intelligence. In the interim, however, machine learning is being applied to a number of cybersecurity problem sets. This article looks more closely at how machine learning is transforming cybersecurity, considering the examples of authentication and masquerade, spam filtering and spam, antimalware and malware, and intrusion detection and intrusion. Machine learning is adding new capabilities for cyber defense and in most cases is useful in conjunction with other approaches. At present, machine learning applications for cyber offense remain primarily proofs of concept.

A flurry of articles has warned that soon cybersecurity will be utterly transformed, becoming a battle between warring artificial intelligences that adapt to each other's moves at "machine speed." Like many claims in an era of artificial intelligence (AI) hype, the image is grounded in ideas of "artificial general intelligence," AI programs that can solve a wide range of problems at least as well as a human. But artificial general intelligence is likely decades away at best—and some say that it is an impossibility.

In the interim, "narrow" artificial intelligence is being used to develop applications for cybersecurity. The current dominant approach in artificial intelligence is machine learning (ML). ML programs use data to develop statistical or probabilistic models that can be used to classify, predict, or generate new examples.

M.A. Thomas is a professor at the College of Information and Cyberspace at National Defense University. She holds a BA in computer and information science from the University of California, Santa Cruz, a JD from the University of California, Berkeley, and a PhD in political economy and government from Harvard University. Her work has appeared in leading academic and policy outlets, including *Foreign Affairs, International Affairs, Defence Studies,* and *Strategic Studies Quarterly.*

This article examines how ML applications are shaping cybersecurity today. It discusses four examples in which machine learning tools are currently being applied: authentication and masquerade attacks; spam filtering and spam; antimalware and malware; and intrusion detection and intrusion. ML is adding new tools to the toolbox, primarily for cyber defenders, but the inherent limitations of machine learning often mean that it is most useful in combination with other approaches.

MACHINE LEARNING

There is no consensus definition of "artificial intelligence," a field of computer science that addresses problems previously thought to require human intelligence. Indeed, the definition is necessarily dynamic as innovation changes expectations of the kinds of problems computers can be used to solve.

Much of the hyperbole about AI is focused on "artificial general intelligence" (also called "general artificial intelligence"), programs that can solve a wide range of problems at least as well as humans can. However, artificial general intelligence does not exist and is not on the near horizon. Some believe it to be impossible.[1] Current AI is called "narrow AI," meaning that the programs are crafted to solve narrowly defined problems.

There are several families of approaches to artificial intelligence and approaches are often combined. For example, "first generation" artificial intelligence, or "expert systems," seek to distill human domain expertise into hard-coded rules that are applied by the program to make the same types of judgments using the same criteria. An example application is the diagnosis of medical conditions.[2] Another has attempted to mimic human reasoning by applying logical rules to data to reach conclusions, such as programs that verify models for quality control.[3] Game-theoretic approaches use game theory to map out possible responses to precisely

defined problems to determine which option is superior. This approach has been used to write programs that win at certain types of competitive games.[4]

However, the current dominant family of approaches in AI is machine learning, which involves building statistical and probabilistic models from data. The models are then applied to new data to make predictions, classify, or to generate new examples from a prompt. The field of ML has benefited from an explosion in the amount of data available for model generation ("training") due to internet and cell phone usage, improvements in data storage, and increased computational power that enables the analysis of that data.

ML models depend on human choices. Researchers select and analyze the input data and consider the desired outcome, deciding how the program will acquire data for model building, what variables ("features") and data should be used for model building, what type of model to build, what algorithm to use to build the model, what parameters should be used in the model building process, and how rare cases will be handled. They curate and provide the training data to build the model, a process called "supervised learning," or they may arrange for data to flow to the program without human curation (for example, from a sensor) in "unsupervised learning" or "reinforcement learning," or they may use a combination of these approaches. The algorithm generates the model from the data, a process called "training" a model. Because the models are probabilistic and statistical, some cases will not be handled correctly. Developers must decide whether some kinds of mistakes are more important than others and what kind of metrics will be used to evaluate model performance. They then test the model on new data and tweak the model or its parameters to optimize performance.[5]

Narrow AI programs are brittle. Novel cases are not handled well because they were not represented in the model's training data. In addition, if the relationship between model inputs and outputs changes, the model may no longer work as well or work at all, a phenomenon called "concept drift."[6] For example, a number of stock market models failed to correctly predict the behavior of the stock market during the COVID-19 pandemic.[7]

Finally, ML models can be defeated. A subfield of machine learning deals with the generation of adversarial examples deliberately designed to be misclassified by a given machine learning model. To generate such examples, researchers need some feedback about how new data are classified. Given that feedback, reverse engineering the model and generating adversarial examples is fairly straightforward.[8]

New tools for cybersecurity continue to be developed and deployed that use expert systems. But research and development of machine learning tools for cybersecurity have exploded in the last five years, in part because of the new availability of data for model training, such as the Github code repository and public malware repositories. Four cybersecurity problem sets highlight the contribution of ML tools: authentication and masquerade attacks, spam and spam filters, malware and antimalware, and intrusion detection.

AUTHENTICATION AND MASQUERADE

A key element of cybersecurity is user authentication, a process used to confirm that a user is who they claim to be before granting access to a system or resource. Authentication factors are *something you know*, *something you have*, or *something you are*. *Something you know* may be a user password; s*omething you have* may be a second device like a cell phone; *something you are* may be your face or fingerprint. ML has enabled wider use of biometric authentication (*something you are*). In a masquerade attack, an attacker seeks to defeat authentication by masquerading as a legitimate user. ML has been used to conduct masquerade attacks by defeating CAPTCHA tests designed to ensure that users are human.

Biometric Authentication

Advances in machine learning have led to increased use of biometrics as means of authentication for many functions.[9] Biometrics can be physiological, such as a person's face, fingerprint, or iris; or behavioral, such as distinct patterns in keystrokes or voice. The biometric information is gathered with appropriate sensors, e.g., a camera for visual information or a microphone for voice information. It is then preprocessed to extract features of interest. These are fed into a classifier model to identify the individual.

Both facial and fingerprint recognition are widely used for cybersecurity. An increasing number of smartphone manufacturers offer the ability to unlock a phone using facial or fingerprint recognition. Both Windows 10 and 11 offer the ability to unlock a laptop using facial recognition with the "Windows Hello" feature.[10] The Government Accountability Office reported in 2021 that sixteen federal agencies have used facial recognition technology for employees to unlock their government-issued smartphones, while two are experimenting with it to authenticate users of government websites.[11] Voice recognition is also used to access some websites.

Biometric authentication is an attractive problem set for machine learning applications because the key features of faces or fingerprints are relatively stable over the period of interest. However, as with all machine learning programs, authentication programs make mistakes. They can also be defeated, including by presenting falsified data to sensors, intercepting and then replaying the identifying data, or using constructed data.[12] Security researchers at Talos reported an 80% success rate in defeating fingerprint recognition, including on cell phones and laptops, using a fingerprint collected from a user and a 3D printer to create a replica.[13] One response has been to attempt to develop ML programs that seek to distinguish between "live" users and static replicas, either by collecting more data through hardware sensors (such as blood pressure) or by software.[14]

Defeating CAPTCHA

CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans Apart") systems are tests designed to determine whether an online user is human in the effort to prevent automated account creation and access to web resources such as webmail forms. They

were initially developed at Carnegie Mellon University in 2000.[15] An example task is to identify a visually distorted number. However, they can be challenging for the visually impaired. Accordingly, many also have an auditory form. Google originally acquired reCAPTCHA, one of the major CAPTCHA systems, as a way to leverage user image recognition to help with its book digitization project.[16]

Researchers have developed proof-of-concept machine learning attacks that bypassed CAPTCHA systems by using ML image recognition on visual CAPTCHAs and freely available machine learning speech recognition utilities on auditory versions.[17] Google released "Invisible reCAPTCHA" the same year. Invisible reCAPTCHA uses machine learning to analyze the user's browser behavior, such as mouse movements, to determine if the user is human.[18] However, this approach relies on cookies to store information about user behavior. If a user used a new device or cleared cookies, the system defaulted to the older reCAPTCHA system, and so was easily defeated. As one security researcher commented, "The fact that Invisible reCAPTCHA can be bypassed isn't because there was a fatal flaw in the design of the newer CAPTCHA. It's that any reverse Turing test is inherently beatable when the pass conditions are known."[19]

SPAM FILTERING AND SPAM

Cybersecurity company Proofpoint estimates from its customer data that less than one percent of cyberattacks depend on system vulnerabilities. The majority depend on social engineering—deceiving people into sharing sensitive information, giving access, or downloading malware.[20] A principal means of conducting such attacks is through "phishing" emails designed to trick users into sharing sensitive information. Verizon's analysis of a sample of convenience of more than 157,000 incidents confirms that phishing attacks and credential theft eclipse attacks that involve the installation of malware.[21]

Phishing emails may be sent in bulk as spam, or they may be specially targeted to a particular user or group of users. Some cyberattacks simply require that the user open the email ("fileless" attacks). Others require the user to open an attachment, which then installs malware. Still others require the user to click on a link that downloads malware or that leads to a website that deceives the user into entering credentials that are then stolen. Phishing emails are designed to persuade the user to take the necessary action to complete the compromise.

To reduce the incidence of this type of attack, as well as general nuisance emails, many email programs use email filters to classify incoming emails as spam or junk, which is then blocked or marked. The earliest spam filters relied on expert systems, hard-coded rules for classification based on an analysis of email traffic. Emails might be screened based on the addresses of senders, their headers, their content, or their language.[22] Spam senders responded by adopting tools that randomize parts of the email message so that they could not be excluded by a hard-coded rule.[23] Later filters used researcher-specified probabilistic or statistical models to evaluate whether an email is likely to be spam.

Machine learning approaches develop the models using data. They benefit from the public release of databases of emails to be used for model training.[24] Email service providers can also draw on the corpus of emails that are sent and received by their users and user classifications of emails. The literature on machine learning approaches to spam filtering is sufficiently voluminous to have given rise to multiple, recent literature reviews.[25]

These approaches are usually combined. Google, for example, uses both rules-based and machine learning approaches to combat spam on its Gmail email platform and claims to be able to detect and block 99.99% of spam, phishing, and malware emails.[26]

It does not appear that spam senders are using machine learning approaches. However, researchers have developed a proof-of-concept spam generator that uses machine learning to evade a spam filter by learning which phrases cause a filter to identify an email as spam.[27] As with all adversarial examples, this requires some feedback regarding how the email has been classified. In this case, the researchers had access to user inboxes and perfect information about how the email was classified.

ANTIMALWARE AND MALWARE

The Creeper was thought to be the first computer virus, designed in 1971 as a proof of concept, displaying the text, "I'm the creeper: catch me if you can!" It prompted the creation of the first antivirus program, The Reaper.[28] Today, malicious software is used to wipe hard drives, steal sensitive information, commandeer computers and equipment or physically destroy them, or to establish "backdoors" for later access.

The use of antimalware software to identify and quarantine malicious executable files on the end user's computer is considered the last line of cybersecurity defense. To operate, antimalware software must be able to classify executable files as malicious or benign. Security researchers would also like to know what the malware does and how it operates so they can check it more effectively. The classic approach to this problem is through signature-based malware detection. Machine learning approaches have been applied to different parts of the problem of signature development and malware detection. They have also been used to identify malware without relying on a fixed signature.

Signature-Based Malware Detection

In signature-based malware detection, security researchers who identify suspicious code study the sample. First, they examine the executable file at rest ("static analysis"). They may reverse engineer it, translating the binary executable code into a higher-level format that allows them to better understand the program logic and actions. They may then run the program to see what it does ("dynamic analysis"). Because of the possibly malicious nature of the code, suspected malware is run in a virtual "sandbox" environment where its behavior can be closely monitored and where it has limited ability to affect the computer.[29]

If the code is determined to be malicious, researchers attempt to identify a "signature," unique blocks or features of the code that can be used to identify it. This signature is added to the database of signatures. The antimalware program scans the files on the hard drive or processes in memory, flagging and quarantining malware with those signatures if it is detected.[30]

This signature-based detection method can only detect known malware for which signatures have been developed but it is still useful. Existing malware is often reused. There is no need for malware developers to develop something new if the old tools will meet their objectives.

However, malware developers do take steps to thwart the signature-development process and signature-based detection. Some malware is written to detect whether it is being run in a sandbox, and, if so, to terminate execution, preventing dynamic analysis. Developers may divide the malware into multiple files, encode or change strings, encrypt the files, or compress ("pack") them to make reverse engineering or detection more difficult.[31] A "stub" executable unpacks and runs the malicious payload. It is not uncommon for malware to have been subjected to multiple layers of encryption and packing. "Fileless" malware avoids writing files to the hard drive where they might be scanned, instead installing in firmware or remaining and running in memory and leveraging native operating system processes to further avoid detection.[32] Fileless attacks have been growing exponentially, bypassing many antimalware programs.

Machine learning approaches have been applied to different parts of this problem set. They include machine learning programs for reverse engineering and for identifying malware without using signatures. At the moment, malware that leverages machine learning remains proof of concept.

Reverse Engineering

One relevant application of machine learning is to facilitate reverse engineering. Computer programs are typically written in a high-level programming language in which a program is expressed as a set of tasks and then "compiled," translated into binary code representing instructions and data for a specific type of computer processor. Humans cannot read and follow binary executables of any length or complexity. Reverse engineering, or "reversing," translates binary code back into instructions that can be more easily understood by humans. Disassemblers translate binary into "assembly language," natural language representations of the instructions to the processor. Decompilers translate binary into a higher-level language program that carries out the same functions. In addition, there are programs that seek to capture and represent the program logic in forms other than a programming language, such as a control flow graph.

Reversing is complicated by the fact that there is not a unique one-to-one mapping between high-level source code and a binary executable. The mapping depends on the compiler, the computer architecture, and code optimization options used during compilation, information that is typically not available to a security researcher.[33] In addition, some information is lost

during the compilation process, such as natural language procedure and variable names that might explain what the purpose of a procedure is, the order in which procedures are called in execution, or information about where functions start and stop.[34]

In addition, developers may take steps to make reverse engineering of their code harder. Malware developers may wish to thwart security researchers. Developers of benign software may wish to protect their intellectual property or prevent tampering. Most commercial software licenses explicitly prohibit reversing for this reason. Examples of these kinds of measures are stripping strings and metadata that would provide clues to program functionality ("stripped" binary executables); using self-modifying code that overwrites its own instructions in memory; and using code that is encrypted at rest and only decrypted in execution.[35]

For these reasons, reversing has been a slow process conducted by a domain expert with software tools that are aids rather than solutions.[36] Disassemblers and decompilers have historically used expert-system, rule-based approaches.[37]

There have been several efforts to facilitate reversing using machine learning. For example, researchers proposed a proof-of-concept machine learning model to decompile small snippets of code from binary, following methods that had been used for text translation.[38] Others built on this approach, achieving greater accuracy by training a machine learning decompiler using code snippets compiled with a given compiler.[39] However, with this approach, each combination of architecture, programming language, and optimization settings would need to be modeled independently.

Other approaches have tackled subparts of the problem of reversing. For example, researchers have used machine learning to determine likely and descriptive names for functions in a stripped executable using control-flow graphs of application programming interface calls.[40] Others have used machine learning for detecting similar sections of binary code or determining where functions start and stop.[41]

Research in this area is still in proof-of-concept phase. It seems likely that machine learning will provide additional tools for reverse engineering but will still fall short of producing a fully automated solution because of obfuscation and information lost in the compilation process and the steps taken to thwart reverse engineering.

Malware Identification without Signatures

Another active area of research is the use of machine learning to classify malware without reliance on hard-coded, manually produced signatures. Researchers build machine learning models of various features of static code or of code in execution.[42] First generation efforts to use machine learning depended on researchers with expert domain knowledge to identify key features from static or dynamic analysis of malware to use as training data. Manually extracted static features include printable strings, application programming interface (API) function calls, function call graphs, control flow graphs, sequences of bytes and opcodes, or the

representation of the patterns of binary code as gray scale images, or a measure of entropy. Manually extracted dynamic features include memory and register usage, instruction traces, and network traffic and API call traces. The second generation uses deep learning to identify the important features from raw code or minimally processed data.[43] Another effort uses the text reports generated by running software in a sandbox.[44]

Machine learning models are more flexible than hard-coded rules and can more easily rely on a wider variety of features to flag suspected malware. However, they will fail to detect novel malware in general and can also be fooled by adversarial examples. Researchers at Skylight Cyber fooled Blackberry Cylance's PROTECT malware detection software simply by appending strings from a popular game to existing malware files.[45]

Environmentally Aware Malware

Although, as of this writing, it is unclear that any malware detected in the wild uses machine learning, researchers have suggested proofs of concept. One example is the use of machine learning for environmentally aware malware.

Environmentally aware malware classifies the environment and executes code based on this classification. This classification has historically been rules-based. For example, the malware may be designed to perform differently or not execute if it detects cues that it is in a sandbox or if it detects a process from a software analysis tool currently running, signals that it is under analysis by security researchers.[46] In 2018, a security researcher reported that 98% of malware that his team analyzed in a sandbox used at least one evasive tactic and 32% used six or more.[47]

Alternately, environmentally aware malware may be targeted, designed to execute its payload only when it is running on a machine with certain characteristics. A famous example of targeted malware, Stuxnet, executed its payload only if conditions were met that described the equipment used for uranium enrichment at Iran's Natanz facility.[48] It sped the centrifuges up periodically while disguising this activity from operators, causing centrifuges to break.

Classification problems are natural candidates for machine learning. Researchers have shown that it is possible to use machine learning, rather than expert systems, to classify the environment. For example, researchers at IBM developed DeepLocker, proof-of-concept malware that uses machine learning to determine whether it has reached the targeted machine and only then generates a decryption key used to unlock the payload, preventing reverse engineering.[49] In their demonstration, DeepLocker triggered only when it recognized the face of a specific user, using the infected computer's camera, another application of biometrics.

INTRUSION DETECTION AND INTRUSION

A final example of the use of machine learning for cybersecurity is in intrusion detection systems. Intrusion detection systems are used to detect unauthorized access to or use of computer systems and networks. They focus on patterns of network usage, system usage, or user behavior.

Intrusion detection systems are classified as signature based or anomaly based.[50] Signature-based intrusion detection is similar to signature-based malware detection. It uses logs of previous intrusions to identify unique patterns of network traffic, which are added to a database. The intrusion detection system then scans for a recurrence of these patterns. Like signature-based malware detection systems, signature-based intrusion detection systems can only detect attacks that have occurred previously and for which signatures have been obtained. Accordingly, they may give "false negatives," failing to detect attacks, even ones that are very similar to previous attacks.

Anomaly-based intrusion detection builds a model of normal system usage and flags deviations. In the 1970s and 1980s, system administrators attempted to detect anomalies by manually examining system audit logs.[51] Expert systems were introduced in the late 1970s. A 1980 report proposed collecting statistical data on system usage by individual users over a period of time, to be used as a baseline, and then using statistical analysis to detect deviations.[52] The first such system was created in the 1980s, using statistical profiles of the behavior of individual users on the system as well as a rules-based expert system.[53]

Although statistical techniques such as regression analysis are now considered to be in the family of machine learning approaches, modern machine-learning, anomaly-based intrusion detection systems rely on a wide variety of approaches, and may focus on different features such as packets, packets over time, packet sequences, logs, or client sessions.[54] Each of these has advantages and disadvantages in terms of the reliability of the outputs.

Unlike signature-based intrusion detection systems, anomaly-based systems can detect novel attacks when they cause a significant deviation from the model of "normal" behavior. At the same time, these systems also flag any other novel pattern, and there are many legitimate reasons why network, system, or user behavior can deviate from previous patterns. Concept drift is also a problem, requiring constant model retraining. Accordingly, these systems can have high false positive rates. One recommendation is to combine them with rules-based expert systems to get the benefits of both.[55] Some hybrid systems use the output of rules-based intrusion detection systems as an input to a machine learning model to attempt to prioritize alerts for human attention.[56]

Researchers are seeking to address challenges in using machine learning for anomaly-based intrusion detection systems, including a lack of data to train models and poor portability of the models to new environments once trained.[57] To be useful, the intrusion detection system must also run without consuming too many system resources so that it does not interfere with regular system use.

Again, given a signal of how an intrusion detection system classifies traffic, it is possible to learn how the intrusion detection system works and develop adversarial examples that evade it. Researchers have developed proof-of-concept intrusions that are successfully concealed in legitimate traffic.[58]

CONCLUSION

More Tools in the Toolbox

While a far cry from the hype of warring, adaptive artificial intelligences, these examples illustrate how machine learning approaches provide new capabilities for defenders in cybersecurity in authentication, spam filtering, and malware and intrusion detection. They are less brittle than rules-based expert systems, providing statistical or probabilistic classifications instead of an absolute classification that requires all conditions to be fully met. The model need not be fully specified by experts. Instead, algorithms can tease complex patterns from large quantities of data. In some cases, machine learning provides novel capabilities, such as biometric authentication.

As has been pointed out in other domains, machine learning is no panacea. Any classifier can be evaded if an attacker has access to enough information about how it classifies and can also shape the input data.[59] Machine learning models cannot reliably classify or predict examples that were not included in their training data, so they are most relevant for problems that do not change substantially over the period of interest. Training models require sufficient good quality data on relevant features of the problem. In the context of cybersecurity, machine learning models must also be nimble and light, providing quick answers without unduly consuming computational or network resources. For these reasons, machine learning tools complement, rather than replace, human experts and existing tools.

To date, most machine learning applications for cyber offense have been proof of concept. There is limited incentive for attackers to invest in machine learning attack tools when the current tools still serve their purposes well. However, as machine learning tools are adopted for antimalware and intrusion detection, it is possible to imagine attackers using machine learning to develop adversarial examples for evasion.

DISCLAIMER

The views expressed in this work are those of the author and do not reflect the official policy or position of the United States Military Academy, the Department of Defense, or its components.

NOTES

1. A 2018 survey of 352 AI researchers asked when "high-level machine intelligence" defined as "unaided machines that can accomplish every task better and more cheaply than human workers" will be achieved. Aggregating their forecasts, the survey found that the researcher estimated a fifty percent chance that this would occur within forty-five years and a ten percent chance that it would occur within nine years. Katja Grace et al., "Viewpoint: When Will AI Exceed Human Performance? Evidence from AI Experts," *Journal of Artificial Intelligence Research* 62 (July 31, 2018): 729–54, https://doi.org/10.1613/jair.1.11222. One point of discussion is the definition of, and the criteria for, "artificial general intelligence." IBM states that artificial general intelligence "only exists today as a theoretical concept versus a tangible reality" given that measures of success have not yet been developed. IBM, "What Is Strong AI? | IBM," accessed December 28, 2022, https://www.ibm.com/topics/strong-ai.
2. See, e.g., A. M. Mutawa and Mariam A. Alzuwawi, "Multilayered Rule-Based Expert System for Diagnosing Uveitis," *Artificial Intelligence in Medicine* 99 (August 1, 2019): 101691, https://doi.org/10.1016/j.artmed.2019.06.007.
3. Roberta Calegari et al., "Logic-Based Technologies for Intelligent Systems: State of the Art and Perspectives," *Information* 11, no. 3 (March 2020): 167, https://doi.org/10.3390/info11030167.
4. John T Hanley, "GAMES, Game Theory and Artificial Intelligence," *Journal of Defense Analytics and Logistics* 5, no. 2 (January 1, 2021): 114–30, https://doi.org/10.1108/JDAL-10-2021-0011.
5. See, e.g., Google Cloud, "Machine Learning Workflow | AI Platform," accessed December 29, 2022, https://cloud.google.com/ai-platform/docs/ml-solutions-overview.
6. Jie Lu et al., "Learning under Concept Drift: A Review," *IEEE Transactions on Knowledge and Data Engineering* 31, no. 12 (December 2019): 2346–63, https://doi.org/10.1109/TKDE.2018.2876857.
7. Will Knight, "Even the Best AI Models Are No Match for the Coronavirus," *Wired,* July 19, 2020, https://www.wired.com/story/best-ai-models-no-match-coronavirus/.
8. Joab Jackson, "Microsoft: Machine Learning Models Can Be Easily Reverse Engineered," *The New Stack* (blog), November 30, 2020, https://thenewstack.io/microsoft-machine-learning-models-can-be-easily-reverse-engineered/.
9. Shervin Minaee et al., "Biometrics Recognition Using Deep Learning: A Survey," *ArXiv:*1912.00271 [Cs], February 8, 2021, http://arxiv.org/abs/1912.00271 (under review).
10. Microsoft, "Windows Hello Face Authentication," July 15, 2021, https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/windows-hello-face-authentication.
11. Government Accountability Office, "Facial Recognition Technology: Current and Planned Uses by Federal Agencies," Report to Congressional Requesters (Washington, DC: Government Accountability Office, August 2021), https://www.gao.gov/assets/gao-21-526.pdf.
12. Battista Biggio et al., "Adversarial Biometric Recognition: A Review on Biometric System Security from the Adversarial Machine-Learning Perspective," *IEEE Signal Processing Magazine* 32, no. 5 (September 2015): 31–41, https://doi.org/10.1109/MSP.2015.2426728.
13. Paul Rascagneres and Vitor Ventura, "Fingerprint Cloning: Myth or Reality?," Talos, April 8, 2020, https://blog.talosintelligence.com/2020/04/fingerprint-research.html.
14. Syed Farooq Ali, Muhammad Aamir Khan, and Ahmed Sohail Aslam, "Fingerprint Matching, Spoof and Liveness Detection: Classification and Literature Review," *Frontiers of Computer Science* 15, no. 1 (2021): 1–18.
15. Stacey Burling, "Captcha: The Story Behind Those Squiggly Computer Letters," Phys.org, June 15, 2012, https://phys.org/news/2012-06-captcha-story-squiggly-letters.html.
16. Rhett Jones, "Google Has Finally Killed the CAPTCHA," Gizmodo, March 11, 2017, https://gizmodo.com/google-has-finally-killed-the-captcha-1793190374.
17. Elie Bursztein et al., "The End Is Nigh: Generic Solving of Text-Based {CAPTCHAs}" (8th USENIX Workshop on Offensive Technologies (WOOT 14), USENIX, 2014), https://www.usenix.org/conference/woot14/workshop-program/presentation/bursztein; Kevin Bock et al., "UnCaptcha: A Low-Resource Defeat of ReCaptcha's Audio Challenge" (11th USENIX Workshop on Offensive Technologies (WOOT '17), Vancouver, BC, Canada, 2017).
18. Rob Verger, "Google Just Made the Internet a Tiny Bit Less Annoying," P*opular Science* (blog), March 18, 2019, https://www.popsci.com/google-invisible-recaptcha/.
19. Nick Flont, "How Cybercriminals Bypass CAPTCHA," Shape Security Blog, July 12, 2017, https://blog.shapesecurity.com/2017/07/12/how-cybercriminals-bypass-captcha/.
20. Proofpoint, "Human Factor Report 2019" (Sunnyvale, CA: Proofpoint, 2019).

NOTES

21. Verizon, "2020 Data Breach Investigations Report," 2020, https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf.
22. Fortinet, "Email Spam Filtering: Different Methods & How They Work," Fortinet, accessed December 16, 2021, https://www.fortinet.com/resources/cyberglossary/spam-filters.
23. See, e.g., Fahim Abbasi, "Tracking the Chameleon Spam Campaign," Trustwave, September 25, 2019, https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/tracking-the-chameleon-spam-campaign/.
24. Emmanuel Gbenga Dada et al., "Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems," *Heliyon* 5, no. 6 (June 1, 2019): 8, https://doi.org/10.1016/j.heliyon.2019.e01802.
25. See, e.g., Dada et al., "Machine Learning for Email Spam Filtering"; S. Abiramasundari, V. Ramaswamy, and J. Sangeetha, "Spam Filtering Using Semantic and Rule Based Model via Supervised Learning," *Annals of the Romanian Society for Cell Biology*, 2021, 3975–92; Aliaksandr Barushka, "Machine Learning Techniques in Spam Filtering," 2020; Alexy Bhowmick and Shyamanta M. Hazarika, "E-Mail Spam Filtering: A Review of Techniques and Trends," *Advances in Electronics, Communication and Computing*, 2018, 583–90; Hanif Bhuiyan et al., "A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques," *Global Journal of Computer Science and Technology*, 2018; Pooja Revar et al., "A Review on Different Types of Spam Filtering Techniques.," *International Journal of Advanced Research in Computer Science* 8, no. 5 (2017).
26. Neil Kumaran, "Spam Does Not Bring Us Joy—Ridding Gmail of 100 Million More Spam Messages with Tensorflow," *Google Cloud Blog*, February 6, 2019, https://cloud.google.com/blog/products/g-suite/ridding-gmail-of-100-million-more-spam-messages-with-tensorflow/.
27. Sean Palka and Damon McCoy, "Fuzzing E-Mail Filters with Generative Grammars and N-Gram Analysis" (Woot '15, 9th USENIX Workshop on Offensive Technologies, Washington, DC, August 10, 2015), https://www.usenix.org/conference/woot15/workshop-program/presentation/palka.
28. Tom Meltzer and Sarah Phillips, "From the First Email to the First Youtube Video: A Definitive Internet History," *The Guardian*, October 23, 2009, sec. Technology, https://www.theguardian.com/technology/2009/oct/23/internet-history.
29. See, e.g., Kurt Beker, "Malware Analysis Explained | Steps & Examples | CrowdStrike," Crowdstrike.com, January 4, 2022, https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/.
30. 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)* (Piscataway, NJ: IEEE, 2019), 1–6, https://doi.org/10.1109/ICCISci.2019.8716381.
31. Trivikram Muralidharan et al., "File Packing from the Malware Perspective: Techniques, Analysis Approaches, and Directions for Enhancements," *ACM Computing Surveys* 55, no. 5 (December 3, 2022): 108:1-108:45, https://doi.org/10.1145/3530810.
32. Sudhakar and Sushil Kumar, "An Emerging Threat Fileless Malware: A Survey and Research Challenges," Cybersecurity 3, no. 1 (December 2020): 1–12, https://doi.org/10.1186/s42400-019-0043-x.
33. H. Xue et al., "Machine Learning-Based Analysis of Program Binaries: A Comprehensive Study," *IEEE Access* 7 (2019): 65889–912, https://doi.org/10.1109/ACCESS.2019.2917668.
34. Yaniv David, Uri Alon, and Eran Yahav, "Neural Reverse Engineering of Stripped Binaries Using Augmented Control Flow Graphs," *Proceedings of the ACM on Programming Languages* 4, no. OOPSLA (November 13, 2020): 1–28, https://doi.org/10.1145/3428293.
35. Shoreh Hosseinzadeh et al., "Diversification and Obfuscation Techniques for Software Security: A Systematic Literature Review," *Information and Software Technology* 104 (December 1, 2018): 72–93, https://doi.org/10.1016/j.infsof.2018.07.007.
36. Daniel Votipka et al., "An Observational Investigation of Reverse Engineers' Processes," 2020, 1875–92, https://www.usenix.org/conference/usenixsecurity20/presentation/votipka-observational.
37. See, e.g., Jakub Křoustek and Peter Matula, "RetDec: An Open-Source Machine-Code Decompiler."
38. Deborah S. Katz, Jason Ruchti, and Eric Schulte, "Using Recurrent Neural Networks for Decompilation," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso: IEEE, 2018), 346–56, https://doi.org/10.1109/SANER.2018.8330222.
39. Omer Katz et al., "Towards Neural Decompilation," *ArXiv:1905.08325 [Cs]*, May 20, 2019, http://arxiv.org/abs/1905.08325.

NOTES

40. David, Alon, and Yahav, "Neural Reverse Engineering of Stripped Binaries Using Augmented Control Flow Graphs."
41. Xue et al., "Machine Learning-Based Analysis of Program Binaries."
42. Daniel Gibert, Carles Mateu, and Jordi Planes, "The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges," *Journal of Network and Computer Applications* 153 (March 1, 2020): 102526, https://doi.org/10.1016/j.jnca.2019.102526.
43. Gibert, Mateu, and Planes.
44. Chani Jindal et al., "Neurlux: Dynamic Malware Analysis without Feature Engineering," *in Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19 (New York, NY, USA: Association for Computing Machinery, 2019), 444–55, https://doi.org/10.1145/3359789.3359835.
45. Kim Zetter, "Researchers Easily Trick Cylance's AI-Based Antivirus into Thinking Malware Is 'Goodware,'" Vice, Motherboard Tech by Vice, July 18, 2019, https://www.vice.com/en/article/9kxp83/researchers-easily-trick-cylances-ai-based-antivirus-into-thinking-malware-is-goodware.
46. Francis Guibernau and Ayelen Torello, *USENIX Enigma 2020 - Catch Me If You Can!—Detecting Sandbox Evasion Techniques* (USENIX Enigma Conference, 2020), https://www.youtube.com/watch?v=_Oes22LvgtI.
47. Siggi Stefnission, "Evasive Malware Now a Commodity," SecurityWeek, May 4, 2018, https://www.securityweek.com/evasive-malware-now-commodity.
48. Marco De Falco, "Stuxnet Facts Report: A Technical and Strategic Analysis" (Talinn, Estonia: NATO Cooperative Cyber Defence Centre of Excellence, 2012), https://ccdcoe.org/uploads/2018/10/Falco2012_StuxnetFactsReport.pdf.
49. *DeepLocker - Concealing Targeted Attacks with AI Locksmithing.* Black Hat, 2020. https://www.youtube.com/watch?v=9g-PllDrJfSU.
50. Ansam Khraisat et al., "Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges," Cybersecurity 2, no. 1 (December 2019): 1–22, https://doi.org/10.1186/s42400-019-0038-7.
51. Jeffrey R. Yost, "The March of IDES: Early History of Intrusion-Detection Expert Systems," *IEEE Annals of the History of Computing* 38, no. 4 (October 2016): 42–54, https://doi.org/10.1109/MAHC.2015.41.
52. James P Anderson, "Computer Security Threat Monitoring and Surveillance" (Fort Washington, PA: James P. Anderson Co., April 15, 1980), https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/ande80.pdf.
53. Yost, "The March of IDES."
54. Hongyu Liu and Bo Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," *Applied Sciences* 9, no. 20 (2019): 4396.
55. Ibid.
56. Ibid.
57. Ibid.
58. Zilong Lin, Yong Shi, and Zhi Xue, "IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection," *ArXiv:1809.02077 [Cs]*, April 23, 2021, http://arxiv.org/abs/1809.02077.
59. Jackson, "Microsoft."