Artificial Intelligence (AI) and the Future of Application Security Testing

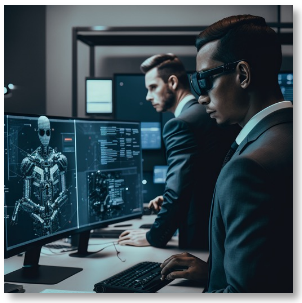
AI revolutionizes how we empower organizations to defend against threats with speed, accuracy, and efficiency.

This white paper offers an overview of the history of AI and its implications, with a specific focus on how Veracode's AI product, Veracode Fix, can transform application security testing.

AI and the Future of Application Security Testing

Artificial Intelligence (AI) is changing our world. In the future, AI will be essential to cybersecurity and application security testing. Organizations require AI to secure assets that power their business. As cyber-attacks become more sophisticated and widespread and leverage AI, organizations will need to rely on AI-powered security solutions to protect their systems and data.

Organizations of all sizes will need to embrace the notion that software can and will change in every step of the software development lifecycle (SDLC). Having a comprehensive security approach that covers planning, design, build, integration, delivery, and production phases of the SDLC will be crucially important in every organization's defense against security risk.



Hackers have embraced AI to unleash attacks on vulnerable software and will do so at an increasing rate in the future. A manual approach to software security will be untenable and organizations will need to embrace an automated security solution with the history and intelligence to identify and automatically remediate risk based on policy decisions.

This white paper will provide an overview of AI technologies, their implications on our future society and the guardrails associated with this technology. We will also discuss how AI is being used to detect and remediate application security risk on the frontlines against criminal and state-sponsored attackers who now have the tools at their disposal to unleash attacks on our most important business assets.

All images in this document were created using AI.

Artificial Intelligence

A Brief History

In the past decade, AI has made significant progress in its development and implementation. Notable milestones include IBM's Watson computer system beating human champions on the TV quiz show Jeopardy! in 2011, Google's deep neural network being used to identify cats in YouTube videos in 2012, and Facebook's DeepFace achieving a 97.35% accuracy rate in facial recognition in 2014.

In 2016, Google's AlphaGo defeated world champion Lee Sedol in the ancient Chinese game of Go. The following year, Google's AlphaGo Zero learned to play Go from scratch and beat the original AlphaGo program without any human input. In 2018, Google's Duplex AI system demonstrated impressive natural language processing capabilities by making phone calls and carrying out conversations with humans.

In 2019, OpenAI's GPT-2 language model generated natural-sounding text with impressive accuracy.

The following year, AI technologies were used increasingly in the fight against the COVID-19 pandemic. For example, by identifying virus hotspots and developing potential treatments. In 2022, OpenAI's GPT-3 language model generated text with unprecedented fluency and accuracy, leading to concerns about the potential misuse of such technology.

The Better Health Generation used CodeBot to generate 91% of the 180K lines of the application code for a mental health application. Forrester reported that ten percent of Fortune 500 enterprises will generate content using AI tools.[1]

AI is becoming increasingly important in the modern world due to its ability to automate and optimize processes, improve decision-making, and enhance customer experiences. The potential of AI is vast and its applications are virtually limitless. As the world continues to explore its capabilities, we are just beginning to comprehend the extent of what AI can do for us.

[1] Forrester Predictions 2023: Artificial Intelligence

The Role of AI

In cybersecurity

While AI has many potential benefits, it is being exploited by hackers to carry out more sophisticated and targeted attacks. Malicious hackers use AI to automate attacks, carry out data theft, evade detection systems, and engage in social systems. For example, AI algorithms are being used to generate polymorphic malware that can mutate in real-time to avoid detection by antivirus software or other security tools.

In more recent years malicious hackers have engaged in social engineering, where they use AI algorithms to generate more convincing phishing emails, voice calls, or text messages. Dark Web forums are busy with discussions on how to use ChatGPT to generate spearphishing emails. A victim's online behavior and social media activity is monitored so that attackers can craft messages that are more likely to succeed in tricking a victim into divulging their personal or confidential information.

The Impact on the Modern Enterprise

In the modern enterprise, automated responses will be the only line of defense.

- Over the next 5 years, global cybercrime costs are predicted to grow by 23% per year, reaching $23.84 trillion annually by 2027. *(Statista)*
- Forecasts show businesses will lose approximately $10.5 trillion in 2025 at an estimated $19,977,168 per minute due to cybercrime. (Cybercrime Magazine)
- Public companies lose an estimated 8.6% of their value after a cyber breach. (Comparitech)
- 45.5% of respondents in a recent survey said that their organization endured between 1 and 5 successful cyber attacks during the past year. (Statista)

Where It All Started

"Attention Is All You Need" by Vaswani et al.,

The Transformer architecture is a type of neural network that was introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017. This model is the foundation of the well-known implementation, Chat GPT. The architecture is based on the concept of self-attention, which allows the model to weigh the importance of different parts of an input sequence when making predictions.

The Transformer architecture has been shown to be highly effective on a wide range of natural language processing tasks, including machine translation, language modeling, and question answering. Its success has led to the development of many subsequent models, including the popular GPT series of language models.

The Transformer consists of an encoder and a decoder, each of which is composed of multiple layers of self-attention and feedforward networks. The encoder takes an input sequence of embeddings and produces a sequence of hidden states, which are then fed into the decoder along with a set of learned context vectors.

Overall, GPT-3 represents a significant advancement over GPT-2 in terms of both size and performance and has demonstrated the potential for language models to generate coherent and human-like responses to text-based prompts.

However, it is important to note that both models have limitations and can generate biased or inappropriate content if not properly trained and evaluated.

Chat GPT-2 and GPT-3 are the largest language models ever created.

The main differences between the two models are:

1. **Size**: GPT-3 is significantly larger than GPT-2, with 175 billion parameters compared to GPT-2's 1.5 billion parameters.
2. **Performance**: Due to its larger size and more extensive training data, GPT-3 is generally considered to perform better than GPT-2. It can generate more coherent and human-like responses to prompts and can even generate original content such as poetry and stories.
3. **Training data**: GPT-3 was trained on a much larger and more diverse dataset than GPT-2, which included a wide range of text from the internet, books, and other sources.

Different Types of AI

An Overview

There are many different types of AI, each with its own set of capabilities and limitations. Some of the most commonly recognized types of AI are:

1. **Rule-based AI**: Rule-based AI uses if-then statements to make decisions. It relies on a set of predefined rules and does not learn or adapt based on new data.
2. **Machine learning (ML)**: ML is a type of AI that uses algorithms to identify patterns and relationships in data. It learns from the data it processes and can improve over time with more data.
3. **Deep learning**: Deep learning is a subset of ML that uses neural networks to identify patterns in data. It is particularly useful for tasks such as image and speech recognition.
4. **Natural Language Processing (NLP)**: NLP is a type of AI that enables machines to understand and process human language. It is used in applications such as chatbots, voice assistants, and language translation.
5. **Robotics**: Robotics involves the development of physical machines that can perform tasks autonomously or with human guidance. It combines hardware and software to enable machines to sense, learn, and act in the physical world.
6. **Expert systems**: Expert systems are AI systems that are designed to mimic the decision-making capabilities of a human expert in a particular field. They are used in areas such as medicine and finance.

These are just a few of the many different types of AI that exist and in this white paper we will focus on Chat GPT, the Transformer architecture and the implications they will have on the future of Application Security Testing.

Chat GPT Benefits

One of the key advantages of Chat GPT is its ability to generate highly natural-sounding responses that are often indistinguishable from those written by a human. This makes it a useful tool for businesses and organizations looking to improve their customer interactions and provide more personalized experiences.

The Next Generation of Application Security Testing

Overall, Chat GPT represents a significant breakthrough in the field of natural language processing. Additionally, it has the potential to revolutionize the way humans interact with computers and digital assistants. Many companies, like Veracode, are leveraging this technology to automate the resolution of application security risk.

Veracode Fix is based on the Transformer architecture which is a type of deep learning model used in natural language processing (NLP) that was introduced by researchers at Google in 2017. It has since become a widely used architecture in NLP, powering many of the state-of-the-art language models used today.

The future of application security testing will be deeply rooted in AI responses to common exploits. As hackers leverage AI to exploit application vulnerabilities at greater frequency, organizations must leverage tools and technologies that enable them to respond quickly, intelligently, and with a set of rules that govern those responses. Veracode’s implementation of AI does exactly this by way of "Veracode Fix".

The Challenges

Public vulnerability databases such as the U.S. National Vulnerability Database (NVD) are inherently incomplete, especially when it comes to open-source packages, which number in the millions. There are many known vulnerabilities that may only be known to the specific open-source project team or even the individual developer.

Veracode has built open-source vulnerability prediction through machine learning to discover these known vulnerabilities that are not yet or may never be in a public vulnerability database. This process detects about 40% more open-source vulnerabilities than there are in NVD. Veracode previously detected vulnerabilities that were eventually published in NVD, but not until months later, giving defenders a much-needed time advantage.

The vulnerability predictor is trained on the information available when a code commit is performed to fix a vulnerability. This information is the code change including comments and the commit comment. With the commit information Veracode can predict whether it is security relevant or not. This predictor can operate on a mass scale daily to scan all open-source repos of packages used by Veracode customers. This gives Veracode customers the opportunity to update packages to remediate their software regardless of whether the vulnerability is in a public vulnerability database or not.

Veracode Fix and Static Security Testing

Almost all AI is only as a good as the data set it has been trained on. Large language models are used to train Chat GPT. In Veracode's case the application security proposed fixes produced by the model have been highly trained, leveraging Veracode's 17 years of data and experience scanning and fixing security vulnerabilities.

Veracode Fix automatically scans developer code for vulnerabilities and proposes solutions to CWEs (a dictionary of software vulnerabilities) and produces highly accurate recommendations. While developers can paste their application code into Chat GPT and ask it to identify security risk (something we don't recommend!) the results are far from satisfactory. The model is capable of identifying basic vulnerabilities but has not been finely tuned with decades of security knowledge like that from Veracode's core data set.

Veracode utilizes GPT large language models as the basis for our implementation of AI with Veracode Fix. GPT stands for "Generative Pre-trained Transformer" and is a type of language model developed by OpenAI and is based on the Transformer architecture.

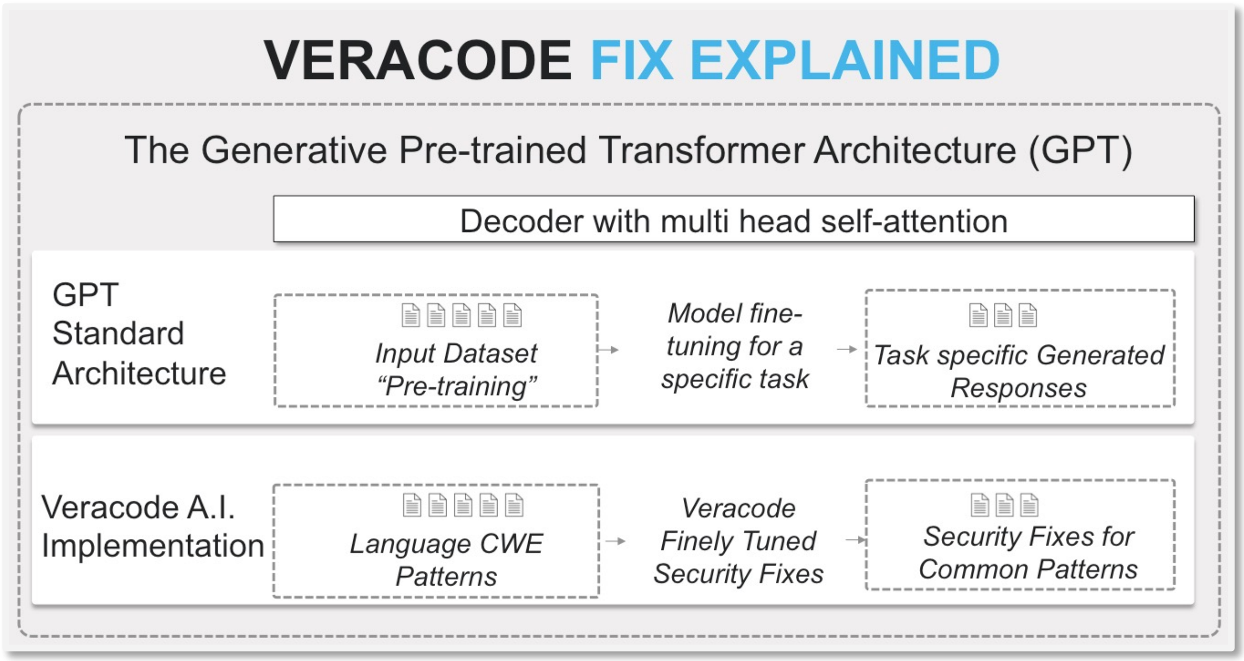
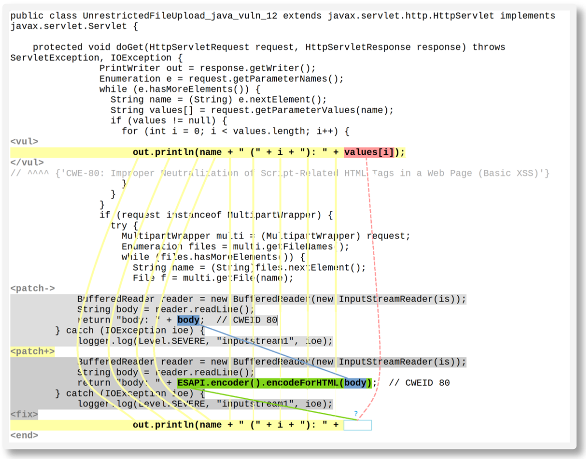

VERACODE FIX EXPLAINED
The Generative Pre-trained Transformer Architecture (GPT)
Decoder with multi head self-attention
GPT Standard Architecture
Input Dataset "Pre-training"
Model fine-tuning for a specific task
Task specific Generated Responses
Veracode A.I. Implementation
Language CWE Patterns
Veracode Finely Tuned Security Fixes
Security Fixes for Common Patterns

Figure 1. Veracode Fix explained

Figure 1 explains the Veracode Fix implementation. Veracode Fix GPT model is a security-specific model that is trained on programming language specific CWE patterns and is fine-tuned on Veracode security fixes.

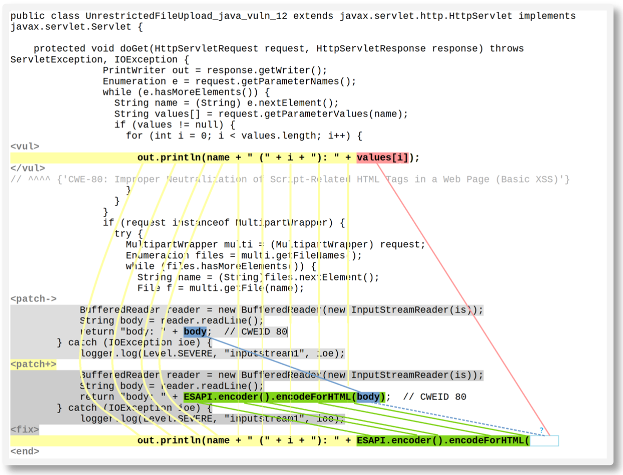
The vulnerable code is placed inside <vul> tag. The Veracode reference security fix is placed in <patch +> tag. The Veracode reference vulnerable code example before reference security fix application is placed in <patch -> tag. The model's output is placed in <fix> tag. The colored connections between different code parts are the attention mappings. The model uses self-attention to focus on the most relevant part of code at any given time. Figures 2- 4 demonstrate Veracode Fix model in the process of fix generation.



```
public class UnrestrictedFileUpload_java_vuln_12 extends javax.servlet.http.HttpServlet implements
javax.servlet.Servlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                PrintWriter out = response.getWriter();
                Enumeration e = request.getParameterNames();
                while (e.hasMoreElements()) {
                  String name = (String) e.nextElement();
                  String values[] = request.getParameterValues(name);
                  if (values != null) {
                    for (int i = 0; i < values.length; i++) {
<vul>
                      out.println(name + " (" + i + "): " + values[i]);
</vul>
// ^^^^ {'CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)'}
                    }
                  }
                }
                if (request instanceof MultipartWrapper) {
                  try {
                    MultipartWrapper multi = (MultipartWrapper) request;
                    Enumeration files = multi.getFileNames();
                    while (files.hasMoreElements()) {
                      String name = (String)files.nextElement();
                      File f = multi.getFile(name);
<patch->
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));
            String body = reader.readLine();
            return "body: " + body;  // CWEID 80
        } catch (IOException ioe) {
            logger.log(Level.SEVERE, "inputstream1", ioe);
<patch+>
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));
            String body = reader.readLine();
            return "body: " + ESAPI.encoder().encodeForHTML(body);  // CWEID 80
        } catch (IOException ioe) {
            logger.log(Level.SEVERE, "inputstream1", ioe);
<fix>
                      out.println(name + " (" + i + "): " +
<end>
```

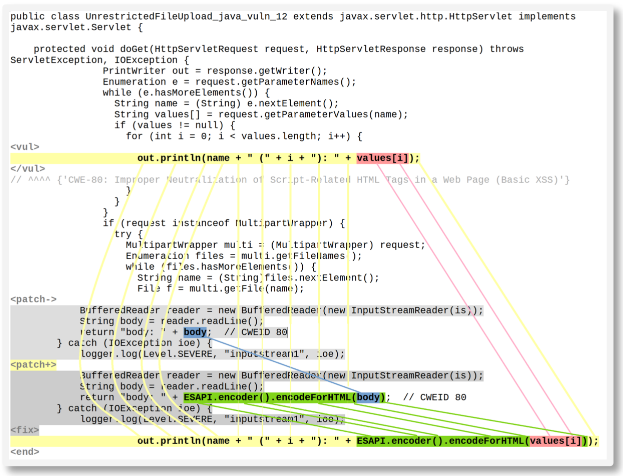
Figure 2. Veracode Fix generating a code fix



```
public class UnrestrictedFileUpload_java_vuln_12 extends javax.servlet.http.HttpServlet implements
javax.servlet.Servlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                PrintWriter out = response.getWriter();
                Enumeration e = request.getParameterNames();
                while (e.hasMoreElements()) {
                  String name = (String) e.nextElement();
                  String values[] = request.getParameterValues(name);
                  if (values != null) {
                    for (int i = 0; i < values.length; i++) {
<vul>
                      out.println(name + " (" + i + "): " + values[i]);
</vul>
// ^^^^ {'CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)'}
                    }
                  }
                }
                if (request instanceof MultipartWrapper) {
                  try {
                    MultipartWrapper multi = (MultipartWrapper) request;
                    Enumeration files = multi.getFileNames();
                    while (files.hasMoreElements()) {
                      String name = (String)files.nextElement();
                      File f = multi.getFile(name);
<patch->
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));
            String body = reader.readLine();
            return "body: " + body;  // CWEID 80
        } catch (IOException ioe) {
            logger.log(Level.SEVERE, "inputstream1", ioe);
<patch+>
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));
            String body = reader.readLine();
            return "body: " + ESAPI.encoder().encodeForHTML(body);  // CWEID 80
        } catch (IOException ioe) {
            logger.log(Level.SEVERE, "inputstream1", ioe);
<fix>
                      out.println(name + " (" + i + "): " + ESAPI.encoder().encodeForHTML(
<end>
```

Figure 3. Veracode Fix generating a code fix



```
public class UnrestrictedFileUpload_java_vuln_12 extends javax.servlet.http.HttpServlet implements
javax.servlet.Servlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                PrintWriter out = response.getWriter();
                Enumeration e = request.getParameterNames();
                while (e.hasMoreElements()) {
                  String name = (String) e.nextElement();
                  String values[] = request.getParameterValues(name);
                  if (values != null) {
                    for (int i = 0; i < values.length; i++) {
<vul>
                      out.println(name + " (" + i + "): " + values[i]);
</vul>
// ^^^^ {'CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)'}
                    }
                  }
                }
                if (request instanceof MultipartWrapper) {
                  try {
                    MultipartWrapper multi = (MultipartWrapper) request;
                    Enumeration files = multi.getFileNames();
                    while (files.hasMoreElements()) {
                      String name = (String)files.nextElement();
                      File f = multi.getFile(name);
<patch->
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));
            String body = reader.readLine();
            return "body: " + body;  // CWEID 80
        } catch (IOException ioe) {
            logger.log(Level.SEVERE, "inputstream1", ioe);
<patch+>
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));
            String body = reader.readLine();
            return "body: " + ESAPI.encoder().encodeForHTML(body);  // CWEID 80
        } catch (IOException ioe) {
            logger.log(Level.SEVERE, "inputstream1", ioe);
<fix>
                      out.println(name + " (" + i + "): " + ESAPI.encoder().encodeForHTML(values[i]));
<end>
```

Figure 4. Veracode Fix generating a code fix

Beyond Static Code Analysis and onto Cloud Native Security

Veracode Fix, in its first implementation will help developers remediate static security findings across all major programming languages. But much like the rest of the AI space, Veracode's use of AI will evolve rapidly to deliver incremental value across the entire SDLC.

In the Future ...

The future of software security will be less about finding and fixing vulnerabilities, and instead focused on preventing security vulnerabilities from ever making their way into the code base and source code repositories. Veracode will lead in these advancements in the following areas:

1. **Prevention**: prevent developers from importing libraries or transient dependencies in open-source libraries that have known vulnerabilities giving security professionals the confidence that new security vulnerabilities are not being introduced through the rapid consumption of open-source software.
2. **Infrastructure-as-code**: intelligent interpretation of code fragments and their potential negative impact on security will be key to securely enabling developers to consume code fragments.
3. **Container Images**: a comprehensive and intelligent detection mechanism will be key to disallowing the adoption of container images that are not secure leading to potential 'all access' exploits when run in production.

These future advancements will be an important step to enable developers to code quickly and securely.

By preventing the consumption and inclusion of OSS, containers images, base operating systems and IaC code fragments that are not secure, Veracode Fix will prevent the most important software security vulnerabilities from ever making their way into an organization's code base.

This will be a huge step forward as organizations transition from scanning, reporting and fixing to proactive preventative development practices.

The Responsible use of AI

Preserving Privacy While Training on Private Data

Some of the best data for training comes from service providers that have lots of private data and have a duty to preserve that privacy for their customers. Veracode has many years and literally trillions of lines of code that has been scanned for vulnerabilities using multiple types of analysis along with manual testing. Most of the vulnerabilities found have been remediated and re-scanned to validate the fix leading to an unprecedented treasure trove of code level vulnerability fix methods. We can leverage this data set while preserving code privacy.

There are a few ways to preserve privacy when using machine learning. One method is to suppress, mask, or anonymize sensitive attributes. In the case of code this would include the names of classes, methods, variables, includes, and other identifiers. Another method is to use cryptographic techniques to share and access data such as differential privacy.

Our research has determined that training machine learning models on reduced and anonymized data work as well as training on private data.

- For us to protect privacy while leveraging organization-specific vulnerability fix knowledge to inform different organizations, the program code is de-identified to remove code that potentially identifies its source or ancestry.
- De-identification is based on the structure of bugs and fixes at the source code construct level based on an abstract syntax tree (AST) or other structural contextual representation of a fix and the corresponding bug.
- Potentially identifying portions of a fix given in its AST are identified and removed or obfuscated without affecting the AST structure.
- This allows one organization to be given the recommendation of a fix that another organization used.

In Conclusion

The Future is Bright

The impact of AI on application security testing cannot be overstated. With Veracode Fix, developers and security teams have a powerful tool that can significantly improve the security of their applications. By automating the identification and resolution of security risks in code, Veracode Fix can save time and resources while also ensuring that applications are secure from the outset.

As we look to the future, it's clear that AI will continue to revolutionize the way we approach technology and security. However, it's up to us to harness its power responsibly and ethically. We must work together to share perspectives on how AI impacts businesses, society, and government regulation, and the potential implications of this technology.

Therefore, we encourage you to connect with us and share your insights on how AI is impacting your business and society as a whole. Let's work together to ensure that AI is used for good and that we can all benefit from its many advantages. The future is bright, and while machines are not taking over, they will undoubtedly be here to stay.

Let's embrace this technology and use it to create a better, more secure world.

Brian Roche
CPO, Veracode

Anna Bacher
Co-Founder & CTO,
Jaroona GmbH

Chris Wysopal
Founder & CTO,
Veracode

VERACODE

Veracode is a leading AppSec partner for creating secure software, reducing the risk of security breach, and increasing security and development teams' productivity. As a result, companies using Veracode can move their business, and the world, forward. With its combination of process automation, integrations, speed, and responsiveness, Veracode helps companies get accurate and reliable results to focus their efforts on fixing, not just finding, potential vulnerabilities.

Learn more at www.veracode.com, on the Veracode blog and on Twitter.