Intel

Next-Gen Security Services with Inline AI Inference

CORPORATE PARTICIPANTS

Lilian Veras
Moderator

Suiqiang Deng
Palo Alto Networks – Distinguished Engineer & Architect

David Lu
Intel – Platform Solution Architect

PRESENTATION

Lilian Veras

Welcome, everyone, to the Intel Network Builders webinar program. Thank you for taking the time to join us today for our presentation titled: "Next-Gen Security Services with Inline AI Inference Powered by Intel Xeon Processor in the Cloud".

Before we get started, I want to point out some of the features of the BrightTALK tool that may improve your experience. There's a Questions tab below your viewer. I encourage our live audience to please ask questions at any time. Our presenters will hold answering them until the end of the presentation. Below your viewing screen, you will also find an Attachments tab with additional documentation and reference materials, including a number of websites and documents mentioned in this presentation.

Finally, at the end of the presentation, please take the time to provide feedback using the Rating tab. We value your thoughts and we'll use the information to improve our future webinars. Intel Network Builders Webinar Series takes place live twice a month, so check the channel to see what's upcoming and access our growing library of recorded content.

In addition to the resources you see here from our partners, we also offer a comprehensive NFV and SDN training program through Intel Network Builders University. You can find the link to this program in the Attachments tab, as well as a link to the Intel Network Builders newsletter.

Intel Network Builders partners have been working to accelerate network innovation by optimizing their solutions on Intel technologies. These industry leaders are recognized in our Winners' Circle Program and Palo Alto Networks is a Titanium Partner. Learn more about our INB Winners' Circle program by clicking on the link in the Attachments tab.

Today we are pleased to welcome Suiqiang Deng from Palo Alto Networks and David Lu from Intel.

Suiqiang has been with Palo Alto Networks for over 10 years, building PAN-OS software and cloud-delivered security services. He's instrumental in bringing the industry's first Inline Deep Learning for threat prevention in the recent Nebula release. With the collaboration with Intel, the team achieved six times speed up in deep learning inference in the cloud. Prior to Palo Alto Networks, Suiqiang worked within Cisco on various embedded networking and security software.

David Lu is a platform solution architect in the Enterprise and Cloud Network division of Intel's Network Platform Group. For the last five years, he has focused on delivering solutions on AI, ML, DL, cybersecurity, cloud-native and confidential computing. Prior to his current role, David worked on various solutions and software architect roles in AR/VR, 3D camera, and wireless.

Welcome Suiqiang and David, and thank you again for joining us today. I will hand over to Suiqiang to start off. Thank you.

Suiqiang Deng

Thank you, Lilian, for the introduction. Hello, everyone. Today, I want to share with you our recent experience in building next-generation security services, and especially delivering Inline Deep Learning to prevent unknown threats, and also how this is powered by Intel Xeon processors in the cloud.

At Palo Alto Networks, we have been at the forefront of innovation in preventing threats to our customers, from traditional signatures to sandbox analysis with Wi-Fi, to the introduction of ML-powered next-generation firewall two years ago. Now, we have taken the next step by introducing Inline Deep Learning technology in our recent Nebula release, with our brand new Advanced Threat Prevention subscription, as well as in advanced filtering and DNS security.

I want to make clear some important differences here first. We're not just talking about machine learning to detect threats. We're not just doing analysis offline or on the box. We're taking deep learning, which is a type of machine learning, specifically well suited to find new and evasive threats, we're putting that inline in the cloud, where we can take advantage of cloud-scale and purpose-built ML processing hardware to prevent advanced threats in real-time.

As many of you know, deep learning is expensive. It's hard to put into the firewall. With new Intel technology in the cloud, now we can. So this is a huge step for our customers in preventing patient zero, stopping new, unknown evasive threats in real-time, and not just file-based threats, but also Command-and-Control, web-based, and other types of threats.

Next, I will walk through how this technology is used in our new product, Advanced Threat Prevention, which gives the industry the first prevention of unknown Command-and-Control in real-time.

Looking at a typical attack lifecycle, Command-and-Control, which is often abbreviated to C2, it's the last opportunity for a network defender to stop an attack before the adversary moves on to data exfiltration or other objectives. Diving deeper into the threat landscape when it comes to C2, there are a few things that stand out. We had noticed an increased use of automation. Adversaries are easily able to spin up and spin down new infrastructure. Attackers are also leveraging public cloud infrastructure to distribute malware and host their C2 server. So defenders cannot just rely on IOC feeds like Ips, URLs. or domains for protection. These techniques simply do not scale. Our own Unit 42 has observed a more than 70% increase in the use of Red Team tools such as Cobalt Strike by threat actors. These tools are purposely built to obfuscate network payloads and emulate real-world traffic to avoid detection. We have also observed the increased use of evasive techniques such as encryption and encoding to bypass traditional security solutions. In reality, many of our customers have not implemented decryption for various reasons, such as privacy or operational overhead. These actors, the threat actors, leverage these loopholes for hiding their C2 communications.

Here are some examples of recent high-profile attacks that leveraged evasive and previously unknown C2 as part of their playbook. The DarkSide Ransomware-as-a-Service group that was leveraged in the Colonial Pipeline attack, researchers observed Cobalt Strike being used as a C2 framework in their operation. Cobalt Strike beacons were also observed in the later stage of the attack in the SolarWinds breach. And lastly, Unit 42 and other researchers have observed Cobalt Strike beacons being delivered as payloads in connection with exploits to the recent Log4j vulnerability.

So looking at the IPS solutions available today, most utilize a signature-based protection mechanism. Signatures are great at what they do. They can prevent known attacks, but they require visibility into malware communications. Once we observed the tactics used by a threat, the research team can develop sophisticated signatures to prevent it in the future. This does not address the problem with patient zero, so signatures are always playing catch up, and they cannot effectively address unknown communication channels. Sophisticated vetting tools such as Cobalt Strike now empower attackers of all skill levels to evade detection. These tools are very easy to use, can be highly automated, and are purposely built to evade signature-based controls by mimicking normal, expected traffic to avoid detection. Most traditional security controls will just see normal network traffic and not suspicious covert communications, allowing such tools to be undetected.

So how must IPS evolve to solve this problem? Not only do we need the ability to stop known attacks. We'll also need the ability to detect unknown attacks and block them in real-time. IPS solutions need to be able to detect these evasive C2 tools and prevent such stealthy behavior with a very low false-positive rate.

These big shifts in the threat landscape have identified some challenges with traditional security controls. With our history of industry innovation, we are uniquely poised to address these changes and evolve IPS to the next level. At Palo Alto Networks, we recently released a new product. Advanced Threat Prevention is a cloud-based Inline Deep Learning service to detect and prevent evasive C2 tactics and techniques in real-time. This new service will initially focus on blocking evasive and unknown C2 traffic that we just talked about. As a cloud-delivered service, we can quickly augment the service to immediately address changes in the threat landscape, with a focus on very high fidelity verdicts, with an automated workflow to greatly reduce any false-positives.

Let's take a look at a diagram that illustrates this workflow. In this diagram, with a compromised user or computer on the left side, malware is trying to communicate with the attacker on the internet with evasive and unknown Command-and-Control traffic. As we discussed earlier, traditional IPS solutions are using signature-based detection to try to block this Command-and-Control traffic. This approach is extremely efficient and works well for known C2 traffic. With malware configured with generic C2 profiles, the C2 traffic will match a signature and be blocked. However, modern C2 tools are designed to allow sophisticated evasive attacks to mimic normal and expected traffic, avoiding detection from traditional signature-based security controls. Some network detection products can provide offline analysis of the traffic using deep learning. With these tools, traffics are mirrored to the offline detection modules on-prem or in the cloud, and the detection is made with techniques including machine learning. However, the analysis is usually slow and cannot keep up with the incoming traffic. Therefore, they can only do after-the-fact detection and remediations.

In our new Advanced Threat Prevention service, we are able to analyze suspicious network traffic using cloud-delivered deep learning in order to provide an immediate verdict on C2 traffic and block the session in-line. Running these new generations of Intel Xeon processors in the cloud, our purpose-built patent-pending models are specifically trained to rapidly find what traditional signatures cannot: a previously unknown or highly evasive covert communication channel. This deep learning analysis is made fast now, and we can put it inline.

We first hold the packet in the firewall and send the relevant information to our deep learning analysis in the cloud, and then the cloud gives back a fast verdict. If the verdict is benign, we'll let the packet go. If the verdict is malicious, we drop the packet. This way, the malicious C2 traffic can be blocked inline in real-time and the attacks will be prevented, and normal traffic will go through smoothly too.

Machine learning and deep learning-based detection engines are only as good as the data that is used in training them. Thankfully, here at Palo Alto Networks, we have the largest malware analysis engine in the industry, WildFire. We see anywhere from five to 10 times the number of files compared to other industry vendors such as VirusTotal. This allows us to leverage a very high fidelity and curated dataset from malware communications observed during cloud analysis, and also collect via Unit 42 threat intelligence, amounting to billions of malicious traffic patterns analyzed over the last decade. With these new detection engines, we have observed a 48% increase in the detection of previously unknown C2 communications that can be blocked inline. We also provide very strong protection, near 100%, for one of the most difficult-to-detect Cobalt Strike C2 techniques in use today. These new capabilities with Threat Prevention subscription was recently released with the PAN-OS 10.2 Nebula software release, and it's made available across the entire network's next-generation firewall platforms.

As we talked about earlier, with Intel's latest Xeon processors and AI software, we were able to achieve fast deep learning analysis in the cloud. Here are some details on this improvement. Intel's recent processors have built-in integrated AI inference hardware. They support AVX-512 and VNNI instruction sets to efficiently handle large amounts of neural network computations. Intel also provides open-source libraries to utilize these instruction sets and provides support for popular machine learning frameworks like TensorFlow. With another open-source software from Intel, the Neural Compressor, we were able to apply quantization to our deep learning models and deploy these models using Intel oneDNN Library. Altogether, these tools give us 6x speed up in inference speed compared to the original deep learning inference implementations.

So, what is quantization? For a deep learning model with a neural network, after training, the weights and the activation functions in the network are by default with 32-bit floating-point precision, or what we call FP32. There is a large number of such weights in the neural network, usually millions or sometimes many more. In a typical CPU, FP32 operations are not handled as efficiently as other operations, and overall inference speed is usually quite slow. To optimize for inference speed, a typical mechanism is to apply quantization to these weights and activation functions. Quantization is to change the weight and activations in a neural network from 32-bit floating-point numbers to 8-bit integers, or INT8 precision.

With this change, INT8 operations can be performed efficiently and in parallel on specialized hardware. This can result in big performance improvements. However, simply changing all the ways from FP32 to INT8 usually lowers the accuracy of the deep learning model. This accuracy loss is often not acceptable, so we need a proper quantization strategy to minimize the impact of accuracy loss. In our collaboration with Intel, we use Intel Neural Compressor software to apply proper quantization to our deep learning models, converting the FP32 models into INT8 models, with minimum accuracy loss.

In this chart, we are showing some test results on the inference speed with our deep learning model. The tests were done in N2 instances in Google Cloud Platform, GCP, with Intel Xeon processors. As you can see in the light blue bar on the left, the 2nd Generation Xeon processor, known as Cascade Lake, the inference speed is almost 50 milliseconds for our FP32 model. With the newer 3rd Generation Xeon processor, also known as Ice Lake, as shown in the dark blue bar on the left, it's about 30% faster, but it still takes more than 30 milliseconds. After applying quantization with INT8 models, as shown on the right side, the inference speed is 6x faster than the original FP32 model. So, on Cascade Lake, it becomes less than 10 milliseconds, and on Ice Lake, it's around five milliseconds.

Next, I will pass the presentation to David to discuss the AI capabilities in Intel products. Thank you.

David Lu

Thanks, Suiqiang. Suiqiang was just introducing Palo Alto Networks' new Advanced Threat Prevention and how it uses Intel technologies to deliver Industry's first inline IPS to stop unknown C2. Now, I want to talk about the Intel Network AI offering.

If you look at the slide, this is the whole stack of the Intel solution. At the bottom is Intel's AI-specific hardware including Intel Atom and Xeon-based CPU, Intel GPU, and Habana AI purpose-built processor. Intel provides two open-source libraries. The first one is Intel oneAPI Data Analytics Library, we also call oneDAL, and the second one is Intel oneAPI Deep Neural Network Library, we also call oneDNN, to unleash the full potential of AVX-512 and VNNI instructions, available on Intel Xeon processors.

oneDNN is for deep learning acceleration. oneDAL is for machine learning acceleration. So, oneDNN is either integration into the deep-learning framework such as TensorFlow, it's already integrated so the people needn't install anything, or we'll have another way. We provide an extension in the deep-learning frameworks such as PyTorch. With the help of this library, the user can easily accelerate their AI training and inferencing.

In addition, Intel also provides Intel Neural Compressor to optimize and quantize the inference model under Intel Architectures. We will discuss those libraries in the following slide.

This is an Intel Xeon Scalable roadmap. Intel continues to lower the computer cost of adopting AI technologies by adding AI acceleration into Intel's processors, without needing exotic accelerators.

Intel first added Intel Deep Learning Boost, we also call Intel DL Boost, featuring VNNI into the 2nd Generation Intel Xeon SP. The code name was called Cascade Lake. It was released in 2019. The Deep Learning Boost got improvement in the Gen 3H Xeon SP, the code name called Cooper Lake, and Gen 3 Xeon SP, code name called Ice Lake.

In the upcoming Sapphire Rapids, which will be released this year, the CPU will be including Intel new instructions called Intel Advanced Matrix Extensions. It's called AMX. Those new expandable two-dimensional registers called "Tiles" are intended as an extensible architecture. The first accelerator implemented is called Tile Matrix Multiply Unit, we also call TMUL. Those innovations will enhance performance for a variety of deep learning workloads.

Now, let's take a close look at why VNNI can speed up the AI inference time.

From the slide, you can find in the 1st Generation Intel Xeon, we need three separate instructions to complete the deep learning computation. In the bottom, you can find starting 2nd and 3^{rd} Intel Xeon, by using VPDPBUSD. That's only one instruction. Those combine three instructions into one. It only needs one single instruction for deep learning computation that formerly required three instructions. This maximizes the user compute resource, it also improves the cache utilization, and avoids potential bandwidth bottlenecks. So, those bring us the significant improvements for deep learning throughput.

Now, we move to the oneDNN, which is like what we talked about before, it's Intel oneAPI Deep Neural Network Library. It can improve productivity under deep-learning frameworks. oneDNN is an open-source cross-platform library to enhance the performance of deep-learning frameworks. It's cross-platform. It can be running on different OS. It can run on Linux, macOS, and Windows. It also supports different types like FP32, FP16, and BF16, and INT8. It's very good to leverage Intel Deep Learning Boost technology, we just talked about it before, like AVX-512 and VNNI. It not only supports Intel CPU but also supports Intel's latest GPU. oneDNN fused operations to optimization performance.

From the right side, you can find three major categories, it's a pain point for the deep learning, like the first one is called Compute Intensive Operations, such as CNN, RNN, LSTM. The second one is called Memory Bandwidth Limited Operations, such as pooling, and batch normalization, and the last one we call Data Manipulation, such as recorder. So, oneDNN, they optimize all those kinds of operations, so you can get the fast performance.

oneDNN replaces compute-intensive standard TensorFlow ops with high optimization custom oneDNN ops. And also, oneDNN does another way. They combine some ops to improve the performance. For example, right now if you do the advanced UI of your team, maybe you use a BERT algorithm, from the right side you can find how oneDNN to fusion the BERT ops, previously to have a lot of ops step by step, you can find it in the big box. And oneDNN combines all this together into just to have "FusedMatMul+GELU". So, all this, what they did, it provides primitive caching to reduce the overhead of calling oneDNN.

The good news is that oneDNN is already integrated into TensorFlow. A user can turn on the oneDNN optimization at runtime in official TensorFlow distributions just by setting an environment variable. This environment variable is pretty easy. Just when you're in Linux, you just export TF_ENABLE_ONEDNN_OPTS=1. That's easy.

After you enable oneDNN, the oneDNN accelerates the inference performance by automatically detecting Intel Deep Learning Boost technology, and the latest supported Intel CPU instructions. So, that means that everything oneDNN takes care of for you. You needn't worry about what kind of instruction set I support, it's automatically detected. If you're running in Intel's 1st Generation, they found that they did not support VNNI, but they did support AVX-512, and they will take advantage of the AVX-512. Or even if you use Intel Atom, they still can take advantage because they detect Atom SSE4.1. That can also improve the performance.

Now, we move to Intel Neural Compressor, because we talked about before oneDNN can significantly improve the performance. However, typically, all those models is FP32. And Intel Neural Compressor can convert FP32 model to the INT8 model, so you can take advantage of Intel Deep Learning Boost technology.

Intel Neural Compressor is also an open-source project. It can quickly deploy low-precision inference solutions under popular deep-learning frameworks.

If you look on the top, it's supporting TensorFlow, PyTorch, MXNet, and ONNX runtime. The good news, it also delivers unified interfaces across multiple deep-learning frameworks for popular network compression technologies, such as quantization, pruning, and knowledge distillation. It supports both Quantization Aware Training and pre-trained model optimization.

And also, Intel Neural Compressor already be involved in a lot of popular communities. They built a very good ecosystem. It's a part of Hugging Face Optimum Project, and contributed quantized models to ONNX Model Zoo.

As we talked about before, by default, most models are using FP32. So, the inference time is going to be a little bit slower, just like what Suiqiang talked about at the beginning, maybe around 40 milliseconds, and you cannot do all the stuff inline, and maybe you need to process all this inference offline. But a user can quantize their model using INT8 to improve the performance. However, sometimes it will affect the accuracy a lot. So, you need to find the best model to balance the accuracy and performance.

How to get the best model? There's a lot of quantization tools in the market to quantize the model. Intel Neural Compressor is recommended for pre-training the model optimization and quantization under Intel Architectures. It can help you to pick up the best quantization scheme to meet the performance and accuracy needs.

Intel Neural Compressor supports automatic accuracy-driven tuning strategies to help the user quickly find the best quantization model. It also implements different weight-pruning algorithms to generate a pruned model with predefined sparsity goals. It supports knowledge distillation to distill the knowledge from the teacher model to the student model.

Here is an example. So from the right picture of this slide, it’s a flow of how Intel Neural Compressor quantized the model.

First, the user needs to prepare three inputs, those are like FP32 model, definitely we want to convert this model, and also we need to have datasets, so we need to find its calibration. And evaluation metrics, so what kind of accuracy you want to get, and how many times you want to try. All those metrics you need to put. Then it will fit into the Intel Neural Compressor. Intel Neural Compressor can have different tuning strategies. You can set them differently, like Bayesian, mse, TPE, or basic.

Then it will query framework quantization capability, which ops can be quantized, which ops can be fused, and query model information, like it contains ops, graph, structure. Then it will do calibrate, quantize, and evaluate loops to find the best model that meets your requirement.

In the left side, you can find how easy to use Intel Neural Compressor to quantize TensorFlow RestNet50 Model. RestNet50 Model, I think a lot of people maybe already know that. So, that’s the reason I use this popular model to give you an example of how you use Intel Neural Compressor.

The left side is the YAML file, which defines some information I talked about before. The first one says, “What kind of framework are you using?” You can put TensorFlow, PyTorch, ONNX. So, in our case, because we tried to use TensorFlow, I put TensorFlow.

Then we put where you put the dataset and also the metrics you want to meet. The right side Python code, it will do the real work for you. It loads the YAML file in the left side and will run the quantization and generate the INT8 model. Then you can use INT8 model, just like what you used in the FP32 model before, so no code change, right? It just repeats this model, add the INT8 model, and you can get the performance jump.

So, here are some references so you can find more info from those links. We already put those links in the meeting attachment part, so you can click all those links and find more information.

Thank you very much for listening. We are ready for any questions you have. Back to Lilian.

Lilian Veras

Thank you, David and Suiqiang, for such an insightful presentation. Really good. We do have a few questions that have come in while you were presenting, so let’s get started on our Q&A section.

Question number one, a member of the audience is saying, “You talked about using Inline Deep Learning to detect and block malware Command-and-Control traffic. What about other types of attacks?”

Suiqiang Deng

Yes, as part of our recent Nebula release, we actually do have other inline machine learning or deep learning for a few other types of attacks, like web content and phishing. And we are adding more. The threat detection is an ever-evolving landscape, and our threat research team, and products, is trying to keep up with all this evolvement in the changing landscape. We are using more and more machine learnings and deep learnings, along with the signatures, and all the other means to stay ahead of our adversaries and protect our customers.

Yeah, thank you.

Lilian Veras

That's great. Thank you, Suiqiang. Another question here, "What were the challenges you faced in putting deep learning inference inline?"

Suiqiang Deng

Yes, good question. I can take that one too. So, deep learning is expensive. With large neural networks and the large amount of ways and activations, it takes time to operate. To put this inline is challenging. The amount of computations is hard to put into the firewall. Otherwise, the firewall itself will be too expensive. So, we are leveraging the cloud. That's why we are leveraging the cloud to utilize the scale and the latest processors in the cloud to achieve this.

And to put… And even with the cloud, to put this kind of analysis inline, we need to be very careful about the impact of user experience. Because most of the traffic is benign, so we want to let them go as fast as possible. To put the compute in the cloud with the latest Intel CPU, and then the software, and the quantization, we can make it as fast as we can, and with multiple global parts, and we can manage this latency introduced to this processing to minimal. So, that will help to keep this processing inline.

Yeah, thank you.

Lilian Veras

Thank you again. We have another question here, I guess, for David this time. David, a member from the audience is saying that he's very interested in using Intel oneDNN. Does he need to rewrite his current codes under TensorFlow, similar to what he did using CUDA API in order to take advantage of Intel AVX-512, VNNI, and how about PyTorch?

David Lu

Yes, this is a very good question. As I mentioned before, oneDNN accelerates inference performance by automatically detecting Intel Deep Learning Boost Technology, and the latest supported CPU instructions. That means for TensorFlow, you only need to enable oneDNN environment variable. You need not change anything in your code. After you enable oneDNN and TensorFlow, you should be able to get significant performance improvement.

For the PyTorch, since it was not integrated Intel oneDNN, it uses the extension. So, you need to add a few line of codes to call IPEX optimization to use Intel oneDNN to improve the inference performance.

Thank you.

Lilian Veras

Thank you, David. One last question that we have. "You talked about how Intel oneDNN can improve the deep learning performance. How about machine learning algorithms, such as XGBoost and Random Forest?"

David Lu

Those questions actually-- because in this webinar, we're only focusing on the deep learning performance improvement, and oneDNN is the hero we are talking about today.

For the machine learning algorithms, another Intel open-source library, Intel oneAPI Data Analytics Library, oneDAL, will help you to improve the performance. It will provide C++ and Java APIs, and also provide Python wrappers, so you can easily use it. Intel also provides extension for Scikit-Learn, accelerates the existing Scikit-Learn code. Also, it will use AVX-512 to maximize performance, such as Random Forest, XGBoost, and LightGBM.

So, that means that when you're using Intel oneDAL, you can get very good performance improvement on the machine learning side.

Lilian Veras

All right, thank you. Well, I would like to thank you both, again, for sharing such great information with us all. I also thank our live audience for joining us today, and ask you to please do not forget to give our team a rating for the live recording, so that we may continually improve the quality of our webinars.

Suiqiang and David, thank you again, and join us next time.

This concludes our webcast today.

Suiqiang Deng

Thank you.

David Lu

Thank you. Thank you, everyone.

Suiqiang Deng

Bye.

David Lu

Bye.