MEET THE PENTAGON'S SOFTWARE MODERNIZATION EVANGELIST

The U.S. Defense Department’s process for updating weapon software is notoriously ponderous, and this threatens to put the country at a disadvantage, if it hasn’t already. What if updates could be developed and dispatched almost continuously to all forces, perhaps even to aircraft in flight? Jan Tegler interviewed the man who wants to make it a reality.

BY JAN TEGLER | wingsorb@aol.com

▲ **The U.S. Air Force** is looking for information from companies that could work with C-130 prime contractor Lockheed Martin to move the aircraft to the DevSecOps method of developing software.

U.S. Army

News of Russia's alleged SolarWinds hacking of U.S. government computers has broken when I get on the phone with Nicolas Chaillan and his public affairs representative to discuss DevSecOps, short for development, security and operations, which is the U.S. Defense Department's initiative to modernize how military software is developed and delivered across the force to improve speed and security. For the Air Force, DevSecOps means in part ending the block approach to updating the software in its aircraft, ground facilities, missiles and satellites. Today, an entirely new version, or block, must be loaded to replace the existing one, and this can be done at best once or twice a year.

In March, the Government Accountability Office criticized the block process and the F-35 program's twice-a-year multi-increment software delivery process as being too slow to modernize the fighter.

"The goal is never to do massive updates," says Chaillan, the French-born software entrepreneur hired by the Defense Department in 2018 to co-lead DevSecOps with the department's chief information officer, currently John Sherman, an intelligence professional serving since January in an acting role. The Air Force, after some months, also made Chaillan its chief software officer, and specifically, the head of the movement toward more frequent updates that was begun by others. "You look at SpaceX," says Chaillan, "they upgrade software 17,000 times a day in a flow of small, incremental changes. Over months it compounds to a significant set of new features."

SpaceX did not respond to my attempts to verify that figure — but it's a colorful way for Chaillan to underscore the dramatic change that he is attempting to orchestrate, a change that has yet to fully take root. The idea is to bury the block approach in the coming years, including for the Air Force's thousands of aircraft, and even enable updates to aircraft in flight.

At stake for the Air Force and other services could well be the ability to stay ahead of adversaries on such critical functions as target recognition, com-

YouTube/Defense Acquisition University

munications and data processing. That said, functions directly related to life and limb, meaning control of aircraft and the safeguarding of nuclear weapons, will be "decoupled" from the shift, Chaillan notes.

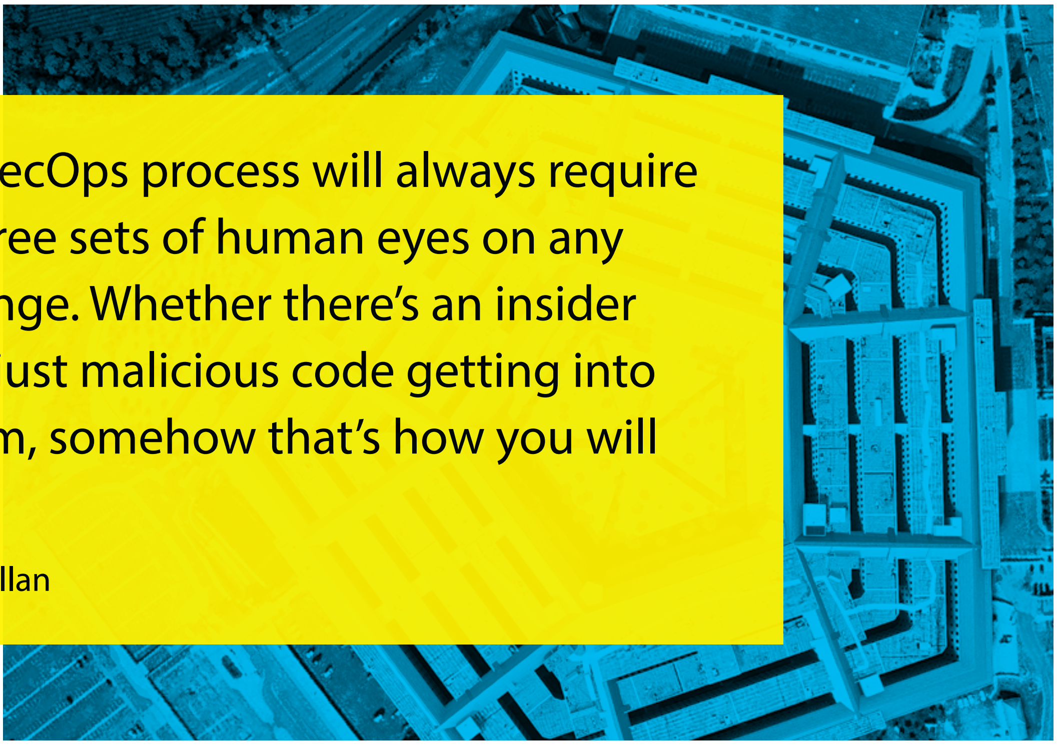
Retiring block upgrades

Right now, much of the military — Chaillan estimates 80% — still upgrades software via the block process, including "the big production aircraft," he says. By 2025, he wants them to follow the SpaceX example of making smaller, more frequent improvements. The legacy block software would not vanish by then, but it would gradually be phased out.

For the military, that would be achieved by empowering its contractors to make software from reusable code and common application tools protected in centralized repositories controlled by Platform One, an Air Force team that wants to manage software development across the services. Software updates would be delivered by a network (defense officials declined to describe it) to computers throughout the military, and by secure datalink to aircraft in flight when necessary.

Last October, the Air Force grabbed headlines by demonstrating the most far-reaching aspect of the concept, delivering over-the-air upgrades to aircraft in flight. Small increments of code were transmitted to a U-2 spy plane flown by a pilot from the 9th Reconnaissance Wing in California. The service reported that the software improved the ability of onboard algorithms to recognize targets for reconnaissance or strikes by other aircraft. Automatic target recognition requires spotting weapons on the ground with visual images, with radar or by the electronic signals the weapons emit. This would require the U-2 to compare its collected signatures to those in a database of known weapons. A U-2 stays up for many hours, and it's possible that in that time it could miss the signature of a new weapon identified elsewhere in the military intelligence apparatus. An in-flight upgrade could catch such a development. Upgrades like this can help the Air Force keep pace with adversaries, Chaillan says.

Still, the first operational examples of in-flight



"Our DevSecOps process will always require two or three sets of human eyes on any code change. Whether there's an insider threat or just malicious code getting into the system, somehow that's how you will find out."

— Nicolas Chaillan

updates could be less consequential. "It might be as simple as changing the color of a button in a cockpit display," he says.

It's also likely that live delivery of code to aircraft on the ground or in flight would be done operationally in "canary releases." In other words, code might be pushed to only a few aircraft of a certain type before an upgrade goes out to every F-35A in the Air Force fleet, Chaillan explains.

"We can send code to a few aircraft initially, establishing hardware-in-the-loop testing to make sure software works as intended, improving capability and not breaking anything. We can also get feedback directly from the users, the pilots and aircrew."

The U-2 flight is an example of the "momentum" that Chaillan says the DevSecOps initiative has gained. After we spoke, for instance, I learned that the Air Force issued a request for information seeking contractors "to transition legacy C-130 software to a DevSecOps infrastructure" that meets "all regulatory testing and cybersecurity requirements."

With such transitions gaining traction, Chaillan spends much of our conversation seeking to defuse concerns that could erode the momentum, SolarWinds perhaps being one of them.

In that hack, Russian operators allegedly managed to inject malicious software into the supply chain for the Texas-based company's popular Orion software. Orion monitors the performance and security of websites, applications, databases and other functions that run on company or government online networks for customers including the U.S. Department of Homeland Security and Treasury Department. As long ago as March 2020, SolarWinds reportedly sent out updates for Orion that unwittingly included malware that left a backdoor for hackers who exploited it to install spyware.

Chaillan brings up the SolarWinds nightmare before I can even ask.

"SolarWinds kind of stuff" requires "eyes on code," he says. "The only way to truly find malicious software today is by human review." In any case, the Pentagon did not need SolarWinds to realize this: "That's why our DevSecOps process will always require two or three sets of human eyes on any code change. Whether there's an insider threat or just malicious code getting into the system somehow, that's how you will find out."

Even so, SolarWinds has, for most cyber experts, vividly underscored the need for human eyes in the loop. Should the Defense Department weaken its emphasis on artificial intelligence and hire more eyes for such tasks as identifying suspicious behavior within the repositories or networks and triggering protective actions? Of course not, he says. For behavior detection, "automation and AI can do much more than any human can do."

Chaillan is not alone in recognizing the need for automation. "A human workforce couldn't keep up with all the work," says Justin Taylor, director of weapons systems engineering at Lockheed Martin Skunk Works in California. The company already uses DevSecOps processes internally and is working closely with the Air Force through a "basic ordering agreement," a contract to provide engineering, software development, cybersecurity and operations, and information technology support for Platform

▲ **The U.S. Government Accountability Office** criticized the Defense Department's program for modernizing F-35 software, which began three years ago and is scheduled to continue into 2027.
U.S. Army

One for an unspecified price. In addition to behavior monitoring, automated testing is part of the Air Force's DevSecOps process at "every step in building an upgrade," Taylor says. Algorithms probe code during development, "making sure upgrades don't break existing apps," he adds.

With Chaillan, I summarize the concerns I'm hearing from cyber experts such as Amir Jerbi, co-founder of Tel Aviv-based Aqua Security, a cloud security firm unaffiliated with the Air Force's DevSecOps initiative. Jerbi fears that if the monitoring and testing are not done exactly right, the military's software updates could be compromised during the process of developing them from code in the repository.

"Then, the minute something is updated you're opening the door to something malicious," Jerbi tells me. "If there is one blind spot in the supply chain, this is where you're going to be attacked."

He wonders what might happen if the code sent in an update to a U-2 acted normally until activated to do something damaging.

Chaillan signals that he's aware of such concerns. He says that the new process, far from increasing such risks, significantly reduces them. For instance, the military once needed a year to fix software. Wherever DevSecOps is fully implemented, an update "could take as little as four hours. That's game changing for security," Chaillan says.

As an example, he notes that the Air Force realized it made a mistake when it ruled that SpaceX could not update the software in its launch vehicles carrying military payloads up until the day prior to a launch, as it does with commercial launches.

"The Air Force launches were less secure than SpaceX's own commercial launches because they had fixes for the software that we couldn't use. We were frozen in time 60 days prior to a launch," he says. "We were actually creating more risk by setting the software baseline than by using the latest update from SpaceX."

Also, security would be built into the code from inception to deployment with multiple layers to prevent or rapidly find and fix anomalies in software. The more automated testing that can be done, the more confidently software upgrades can be made, Chaillan maintains, explaining that if a functionality or security issue is found in a software upgrade, a new test can be created immediately to prevent it from occurring again.

"Fail fast but don't fail twice for the same reason" is Chaillan's motto.

Genius or new vulnerability?

To speed up and safeguard software creation, developers get access to a range of approved Platform One development tools and a set of department-vetted programming languages and databases, plus access to Iron Bank, one of the repositories.

Iron Bank stores discreet pieces of open source, commercial code and also custom code created to meet Pentagon requirements in containers, the cloud-computing term for discrete packages of software that hold pieces of code needed to create or run software applications.

All containers in Iron Bank are hardened, meaning the code inside them has been reviewed and accredited for use by the Air Force's Office of the Chief Software Officer. Developers can contribute new code to the repository, expanding the library of containers. Chaillan likes to refer to them as Lego blocks, explaining that the Pentagon wants developers to assemble code from Iron Bank's containers for applications and upgrades for everything from aircraft to ships to electronic warfare sensors.

I give Chaillan the short version of the long list of reservations that cyber experts have expressed to me.

Jerbi worries about a "unique, crafted attack" in which a nation-state will try to find a supplier — a company within the defense industry that creates new code to be contributed to Platform

▲ **Officials at the Air Force's U-2 Federal Laboratory** participate in a computing experiment. Days later, the lab proved that the service can update a U-2's software while it's in flight.

U.S. Air Force

One's repositories — "that uses a certain component, let's say a Java library, and attempts to poison that library. This will be their Trojan horse into the environment."

The poisoned code might "slowly, methodically move laterally through Platform One," affecting the software build process, making its way into containers meant for upgrades.

"You will select a container," Jerbi says. "You will insert a piece of code" for a sensor upgrade, for example, "that looks and acts normal as legitimate software, connecting to whatever components it should or needs to connect with, meeting whatever conditions it needs to meet."

Once sent to the mission computers onboard an aircraft, the malicious code in the container "will find small holes enabling it to add or subtract data where possible," Jerbi explains.

Then, "when [an aircraft] is connecting to a cloud or a network to download the next upgrade or code version, it might add a few more pieces of data to the new services or it might reveal an additional few bytes of data."

Malicious code in a container could alternately exfiltrate data, directing a file to be downloaded to a container an adversary has access to. That would be a veiled way to gather intelligence on a system or operations.

"As we've seen with SolarWinds and as our own research team is seeing with containers, there are active measures being taken to create backdoors through the supply chain in a way that is not detectable with simple signature-based or vulnerability testing," Jerbi says.

He's not saying this will happen. Testing and monitoring will be the keys to going beyond "run of the mill hygiene," he says. If enough testing can be done "with nothing that hasn't been approved or tested moving to deployment" then continuous updates "could be successful in theory," Jerbi says.

Another expert, Tamer Hassan, the founder of WhiteOps, a New York City-based cybersecurity firm and a former Air Force C-130 and HH-60G Pave Hawk pilot, puts the stakes in human terms.

"If bad data got into a navigation system on a Pave Hawk, that could spoof position information or interfere with the mission you do, combat search and rescue." Would he be confident flying a Pave Hawk helicopter that received a software upgrade in flight?

The answer might be surprising: "Absolutely," he says, "if I knew that if X, Y or Z breaks because of an upgrade, there's a plan — that any problems, like a bug pushed to the navigation system, can be mitigated fast."

Still, the overall message from cyber experts is one of caution and perhaps nervousness. Is it wise to collect code in centralized locations this way?

Chaillan does not waffle at all. "If you ask the same thing of the Secret Service about the White House, they're going to tell you that centralizing all the risk into a central place" makes it "easier to secure the White House," Chaillan says.

True, Platform One is "the crown jewel of the department," and "if someone were to get access or get inside of Iron Bank" — he compares it to someone jumping the White House fence — "that's a big problem." And it could well happen: "Nothing is perfect." But the possibility of a breach doesn't mean that centralization is not "the right thing to do."

Like the Secret Service, he is not just relying on perimeter defense alone. "At the end of the day, it's always scary to put all your eggs in the same basket, but at the same time, our basket in Platform One is sort of diversified," he explains. "We're not locked into a single product or a single tool. We always have a lot of options."

Also, "centralizing talent and execution in Platform One" enables the Defense Department to "do a much better job of securing software for rapid upgrades as a unified team than doing it in a vacuum" with teams in industry and the military working separately. And DevSecOps requires the testing and monitoring that Jerbi and other experts highlight as crucial.

As for the containers, "If a bad actor gets into container A, and it tries to send a command to get into container B, and it has never tried to do that before, our behavior detection system will detect that this isn't normal and will kill container A, alert us and restart container A back to immutable state."

The practice is part of a strategy known as "moving target defense" wherein continuous software updates yield an ever-changing target, Chaillan adds. "That makes it very difficult for any bad code to remain in the system for a long period of time."

Here is Chaillan's bottom line about security: The DevSecOps cybersecurity "is probably the most advanced on the planet." Activating malicious code will be extremely difficult, and if an application acted abnormally, it would be detected and stopped or patched fast.

Trust nothing and encrypt

Last year, the Air Force and the rest of the Defense Department began to adopt a cybersecurity strategy known in the software industry as Zero Trust, so called because it assumes that no request from outside or inside a network like Platform One is trusted at the start. Every action must be challenged and authenticated.

Chaillan describes Zero Trust as "denial by default," adding that it can prevent malicious code from migrating through containers and stop any adversary from trying to penetrate Platform One via the cloud computing network that hosts it.

The starting point is that no piece of code or application in a container can communicate with or make a request of another piece without approval.

"If I want to create connectivity between two things — two containers — I have to 'white list' or approve that traffic. That's one pillar of Zero Trust," he says.

The second pillar is encryption. Any time a container communicates with another container to execute a service, the data will be encrypted, according to Chaillan. "No data will ever be transferred in the clear," he says, such as when a new container is sent to an aircraft in flight.

The third element of Zero Trust is a certificate authenticating communication between containers. "The software managing containers issues a short-lived certificate, or identity if you will, so we know container A is container A and not container B pretending to be container A," Chaillan says.

Each container gets rotating identities that expire as often as every hour. Even if a hacker "managed to steal one of the identities to use to authenticate, that identify will expire quite soon."

Zero Trust also applies to the cloud network underpinning Platform One. All developers must go through a single access point.

A series of Zero Trust protocols protect Platform One, challenging every request, even when a user has gained access into the platform. Any request to pull code from the repositories or make use of security or development tools is fully authenticated, authorized and encrypted before access is granted. This "microsegmentation" also prevents lateral movement, Chaillan says.

Software and defenses can never be perfect, Chaillan says. But because adversaries are developing capabilities faster than the U.S., "the risk of not continuously updating software, even in flight, is greater than the risk of not doing it." ★

Ben Iannotta contributed to this story.

▲ **Software on a U.S. Air Force U-2 was updated** while the aircraft was in flight, a method that can help the Air Force keep pace with adversaries, says Nicolas Chaillan of the Defense Department.
U.S. Air Force