timesys

A Timesys E-Book

Vulnerability Management for Embedded Systems Using Open Source

September 9, 2021 - Version 3.0

Vulnerability Management for Embedded System Using
Open Source — A Timesys E-Book

© 2020 Timesys Corporation
Corporate Headquarters
1905 Boulevard of the Allies
Pittsburgh, PA 15219
UNITED STATES
T: +1.412.232.3250
Toll-free: 1.866.392.4897
www.timesys.com

Contents were previously published by Timesys Corporation
in on-line blog and web site publications in 2019 and 2020.

For more information about Timesys and security for
embedded open source systems, please visit us at
www.timesys.com.

Contents

04

01 | Introduction:
Managing Embedded System Product Cybersecurity Risk

11

02 | CVE Monitoring & Management:
Guidance on Best Practices

17

03 | Process Template:
Managing Security Maintenance in the Product Lifecycle

21

04 | Timesys Vigiles:
Vulnerability Management & Mitigation for Stronger Embedded System Security

01

Managing Embedded System Product Cybersecurity Risk

The security of products you bring to market can no longer be a secondary consideration.

Hundreds of new software vulnerabilities are announced every week by vulnerability tracking services such as the listings of Common Vulnerabilities and Exposures (CVEs) in the National Vulnerability Database (NVD). The NVD is maintained by the US National Institute of Standards and Technology (NIST).

End-user customers are increasingly cautious about deploying products that might expose them to vulnerability exploits. Security controls, update procedures, patching, regulatory compliance and security standard compliance now rank high in the requirements issued by customers seeking to buy and deploy embedded systems.

These two trends mean that makers of embedded system products need to take an active approach to assessing and mitigating security problems in software in products they develop. Device makers need to focus on security when products are designed and when they are managed throughout the product lifecycle.

Industrial control systems, medical devices, automotive systems, Internet of Things (IoT), Industrial Internet of Things (IIoT), smart devices and many other embedded systems products must be secure. The software running on these devices needs to be protected from exploits that can compromise system integrity, put confidential information at risk of exposure, or impact system performance. In some cases, device security failures can have a direct impact on the health or safety of users.

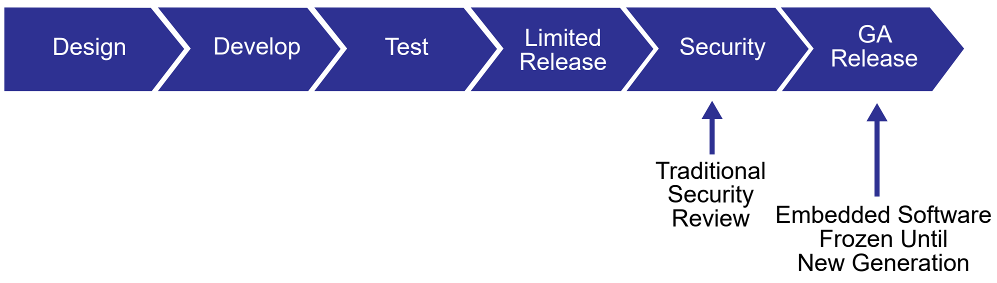
This e-book details important trends in embedded system software security, both in how system software is designed and built to be secure, and how security is maintained after products are released to customers.

An essential consideration is how you, as a product development organization, assess and manage security risk across your product lines.

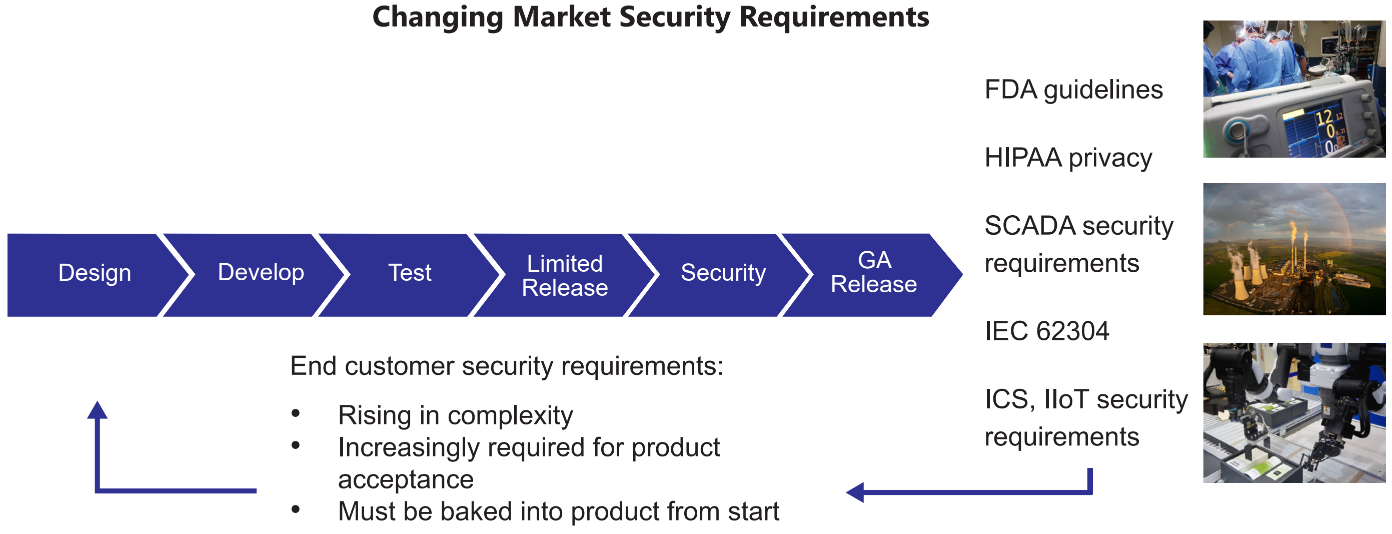
Security in the product development process

In the traditional software development lifecycle (SDLC), security is often a release gating assessment. It's a review conducted when proposed final products are being evaluated for release to market.

Security's Traditional Position in Software Development Lifecycle


Design
Develop
Test
Limited Release
Security
GA Release
Traditional Security Review
Embedded Software Frozen Until New Generation

Changing Market Security Requirements


Design
Develop
Test
Limited Release
Security
GA Release
FDA guidelines
HIPAA privacy
SCADA security requirements
IEC 62304
ICS, IIoT security requirements
End customer security requirements:
• Rising in complexity
• Increasingly required for product acceptance
• Must be baked into product from start

Much like Quality Assurance on a final release candidate, such security assessments evaluate the release candidate's compliance with security requirements and highlight any deficiencies for correction before product release sign-off.

After the security review was passed, the product software would be released to the market. Generally speaking, embedded system software would be frozen at that point until the next generation was developed and released.

This traditional approach to security creates three problems:

1. The world is not frozen, even if your software is. Besides the explosion in vulnerabilities and exploits in recent years, many other aspects of how your products are deployed and used will have major impacts on the security of these systems. Embedded system products are increasingly connected to networks for management, monitoring and maintenance. Remote updates and patching are widespread requirements. Updates to the Linux kernel and to third-party components may occur that require security maintenance for your product.

2. The traditional approach to security at the end of the SDLC means that software is inflexible and unable to be adapted as security requirements evolve over time. Customer and market security requirements are ever changing and are often critical to customer acceptance. Each industry and application may have a unique set of security requirements to meet and new ones may be imposed at any time as the patterns of security vulnerabilities and exploits change. Creating an architecture that meets these requirements, and allows adaptation to new ones, should be considered at product design stages.

3. Competition and market changes are compressing development cycles. The rise of IoT, IIoT, data analytics architectures, smart devices and other market changes are increasing pressure to bring products to market faster. This means development cycles are being compressed, often by adjusting product scopes or using third-party software such as open source components to cut development time. This means that the traditional security review at the end of the development process can become a bottleneck that slows time to market.

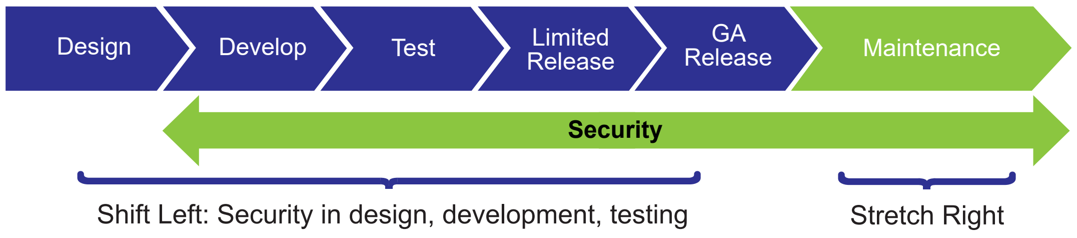
Security Expands: *Shift Left and Stretch Right*

These factors mean that security needs to change from taking place as a single review and assessment at the end of development prior to product release.

Instead, security must become a continuous, developer-driven process at every stage of product development and continuing after product release.

In development process terms, this means that security needs to "shift left" and move earlier in the SDLC. At the same time, it needs to "stretch right" as a continuous process that extends beyond release throughout a product's maintained lifetime.

Security's Expanded Role in SDLC



This concept of expanded security across the SDLC has several important implications for your product development and maintenance processes:

- Security tools must be aligned with development workflows and environments.
- Highly accurate vulnerability identification is needed at all SDLC stages, across all versions, all components, and all branches of your product's software.
- Your development teams should build products using the latest, most secure third-party components.
- Your product maintenance process should include continuous vulnerability monitoring,filtering and triaging, along with tracking availability of fixes and a quick mitigation process.
- Obtaining accurate vulnerability data and cutting the chance for false positives can minimize impact on your development team.

As these points imply, security can and should be turned into a proactive part of your product development and maintenance process, as opposed to a reactive activity that takes place only when things go wrong.

Making the change to proactive security will have the effect of decreasing overall risk across your product lines.

Managing product security risk

Today's most advanced product development organizations focus on lowering the security risk across their product lines, to decrease the chance that a vulnerability will be exploited in a deployed product.

The process for managing product line security risk considers these factors:

- What is the level of risk of a vulnerability exploit you will accept?
- What is the effort and cost to assess the exposure to known vulnerabilities?
- What is the level of accuracy of the exposure level?
- What is the effort and cost to mitigate identified vulnerabilities?
- How quickly can you mitigate identified vulnerabilities?

Determining the level of risk that your organization will accept involves many factors that are specific to your market, your product strategies, your customers' requirements and so on.

At the simplest level, a known, exploitable vulnerability that is present in products across your entire customer base represents the highest level of risk. Unless there are mitigating circumstances around deployment modes, there are likely very few companies that will consider this an acceptable level of risk.

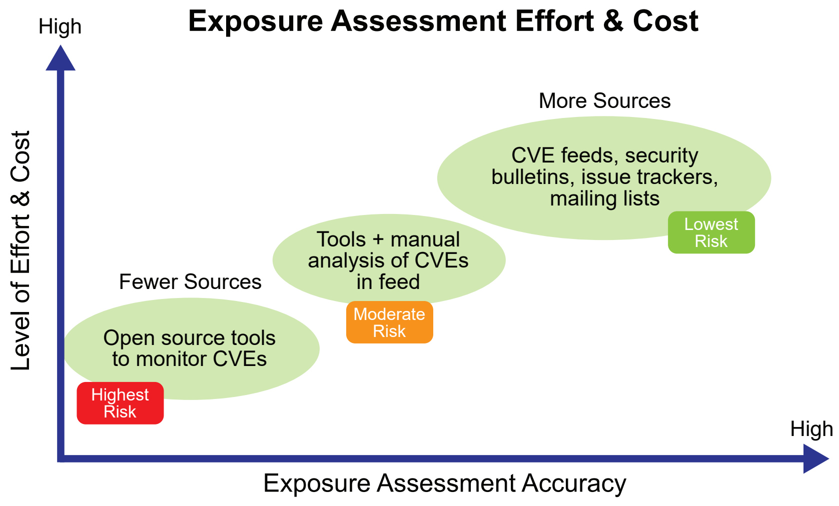
The opposite extreme is the elimination of all exposure to all known exploitable vulnerabilities across your products deployed in all your customers. While this lowest level of risk seems to be an ideal case, the level of effort and cost to achieve it could be extreme. Cost-benefit analysis can help to determine the right balance once you hit a point of diminishing returns.

Determining your strategy and level of investment in assessing vulnerability exposure should consider two main factors.

First is the level of accuracy of the exposure assessment to known vulnerabilities compared to the level of effort required.

Second is how quickly you can achieve mitigation of a known vulnerability compared to the level of effort to more rapid mitigation.

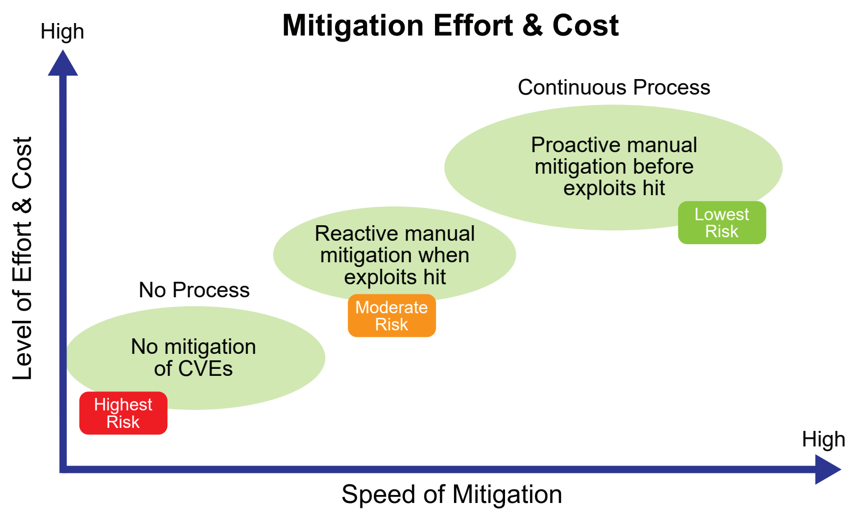
The following charts illustrate the correlation between level of effort and better results.


Exposure Assessment Effort & Cost
High
Level of Effort & Cost
More Sources
CVE feeds, security bulletins, issue trackers, mailing lists
Lowest Risk
Tools + manual analysis of CVEs in feed
Moderate Risk
Fewer Sources
Open source tools to monitor CVEs
Highest Risk
High
Exposure Assessment Accuracy

In the curve for Exposure Assessment Effort and cost, the lowest level of risk is achieved when you incorporate more sources of vulnerabilities data into your assessment and employ tools to assist with analyzing, tracking and filtering vulnerabilities.

But as the curve shows, higher accuracy comes at a higher cost and level of effort.

Similarly, a security best practice is to mitigate identified vulnerabilities as rapidly as possible. As shown in the next chart, the highest level of risk occurs when you devote no resources to mitigating vulnerabilities. The lowest level of risk is achieved when you assign teams to analyze and mitigate identified vulnerabilities before an exploit is released in the wild. In other words, the more proactive the mitigation process, the lower the risk of an exploit and breach.



But mitigation speed demands ever-increasing investment of effort and cost to drive risk lower.

Achieving lower risk at lower cost

In our work with embedded system product developers around the world, we have seen four best practices that enable development teams to jump the curves illustrated above and achieve lower risk at a lower level of investment.

These best practices are:

- **Automated Software Composition Analysis and SBOM generation**

  Accurately determining the level of exposure to a given vulnerability requires that you first accurately understand exactly what is in your product. A Software Bill of Materials (SBOM) is basically an inventory of your software's components.

  Virtually every product on the market today contains a great deal of code sourced from third parties. In many cases, these are open source components that development teams find and integrate into their products to reduce the time spent on developing common features and functions.

  But this practice means it can be a challenge to fully understand exactly which components and which versions are integrated in the product.

Generating and managing your SBOMs can be a very time-consuming manual task that drives up the level of effort for security monitoring. But Software Composition Analysis (SCA) tools are now available that can automatically analyze your software and generate an SBOM. The SBOM in turn becomes your guide for analyzing reported vulnerabilities to determine if any of your software components are exposed.

- **Automated, augmented feeds and filtering**

  As illustrated in the Exposure Assessment chart, more accurate assessments are produced when you incorporate more vulnerability feeds and reports into your monitoring process.

  But it can become very costly to manually monitor the many available sources and then filter these lists to pinpoint only those that pertain to your product's SBOM.

  The best practice is to automate the monitoring and filtering process as much as possible. That means establishing a continuous process for taking in vulnerability feeds, normalizing the outputs, and comparing them to the project SBOMs your team is working on. Then your team is notified when a vulnerability hit occurs. The result is much more efficient monitoring and analysis because your team can focus on only those vulnerabilities that matter.

- **Collaboration & sharing across teams**

  A developer-driven security management process is the most effective method for ensuring security is baked into products from the outset and maintained after release.

  To allow development teams to most efficiently address vulnerabilities, their work on exposure assessment and vulnerability mitigation should be built around a collaborative process and tools that permit easy communication, documentation of findings and tasks, and sharing of analysis and mitigation actions across teams and across projects.

- **Automation-assisted analysis & mitigation steps**

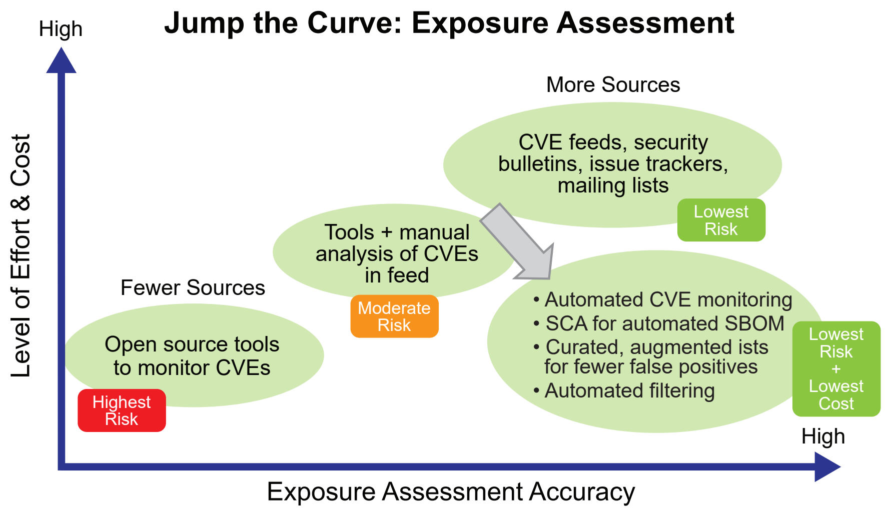
  Similar to using automation for software analysis and vulnerability filtering, a development team's analysis of a given vulnerability and potential mitigation of it can be made much more efficient with automated reporting of available patches and updates.

  For example, a given third-party product component may be affected by a particular vulnerability. An automated scan of the releases of the components listed in the SBOM can flag a release that addresses the vulnerability. The development team immediately knows how to mitigate the vulnerability and has greatly accelerated mitigation speed without expending additional effort.

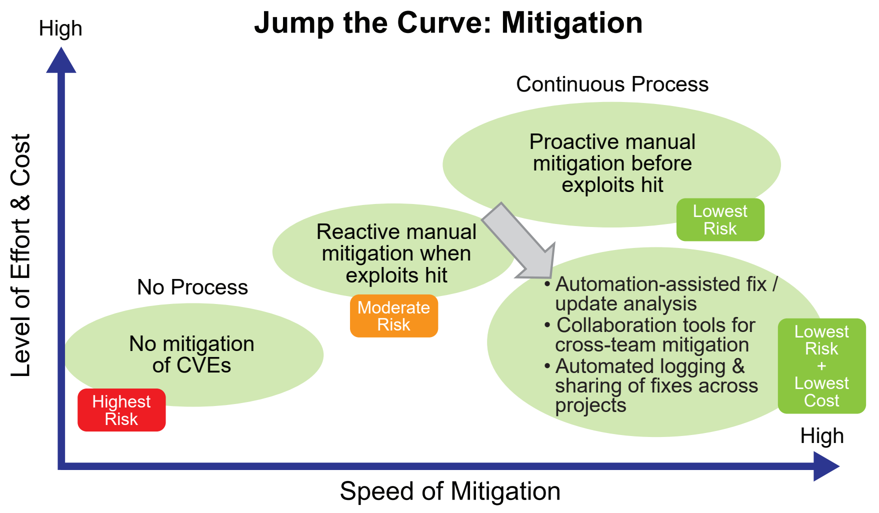
Jumping the curve in exposure assessment and mitigation speed

Employing the four best practices described above will enable your security management process to break the correlation between level of effort and better results.

The chart below illustrates how using automated SCA tools, automated filtering and similar steps to augment monitoring across more vulnerability reporting sources can enable your process to "jump the curve." The result is better exposure assessment for lower risk at a lower level of effort and cost.


Jump the Curve: Exposure Assessment
High
Level of Effort & Cost
More Sources
CVE feeds, security bulletins, issue trackers, mailing lists
Lowest Risk
Tools + manual analysis of CVEs in feed
Moderate Risk
Fewer Sources
Open source tools to monitor CVEs
Highest Risk
• Automated CVE monitoring
• SCA for automated SBOM
• Curated, augmented ists for fewer false positives
• Automated filtering
Lowest Risk + Lowest Cost
High
Exposure Assessment Accuracy

The same benefits can be achieved with mitigation as shown in the chart below. You can drive risk lower at lower cost by establishing automation-assisted mitigation analysis and a collaborative security management process.


Jump the Curve: Mitigation
High
Level of Effort & Cost
Continuous Process
Proactive manual mitigation before exploits hit
Lowest Risk
Reactive manual mitigation when exploits hit
Moderate Risk
No Process
No mitigation of CVEs
Highest Risk
• Automation-assisted fix / update analysis
• Collaboration tools for cross-team mitigation
• Automated logging & sharing of fixes across projects
Lowest Risk + Lowest Cost
High
Speed of Mitigation

The process enables your team to move from a reactive approach to vulnerabilities, scrambling to respond when exploits hit, to a proactive approach that enables customers to be protected typically before an exploit is available.

The net result of introducing automation into your exposure assessment and vulnerability mitigation process is a significant reduction in product security risk without causing costs to explode. This in turn means your customers will have less exposure to vulnerability exploits and will be able to rely on your products as secure.

02

CVE Monitoring & Management: Guidance on Best Practices

Timesys' Director of Engineering, Akshay Bhat, presented a session on Open Source Security at the Embedded Linux Conference North America 2019. For this Q&A interview, Timesys VP of Marketing Adam Boone asked Akshay to share his views on the challenges and best practices for maintaining security in Open Source Embedded System products.

Adam Boone: Why should product developers and engineering managers be familiar with CVEs and make an effort to monitor them?

Akshay Bhat: I think everyone recognizes it is important to bring products to market that are secure and that stay secure throughout their deployment lifecycles.

Who wants to deliver a product that is going to put a customer or user at risk of a data breach, system hijacking, or other security issue now or two years from now?

This is especially important today as embedded systems products are being deployed in greater numbers than ever before, and in locations and configurations we have never seen before. Plus, these systems are supporting critical processes and essential services we all depend on. Think about the Internet of Things, Industrial Internet of Things, industrial control systems, utility control systems and so on.

And perhaps most critically, more and more of these embedded systems contain Linux and other open source components. While open source is a fantastic way to get products to market quickly, you need to make sure these components are secure so that customers are not put at risk.

So anyone working on the software for an embedded system product today needs to know about CVEs that are announced and maintained in the NVD.

The challenge, of course, is how much effort you can expend in monitoring, managing and fixing vulnerabilities if they happen to affect your products.

While many of us know about the NVD, there actually are many other potential sources of important vulnerabilities, such as manufacturer security bulletins, security mailing lists, and bug lists. And there are what you might call silent bug fixes in which a manufacturer fixes a bug without issuing a public notification.

"And perhaps most critically, more and more of these embedded systems contain Linux and other open source components. While open source is a fantastic way to get products to market quickly, you need to make sure these components are secure so that customers are not put at risk."

In a perfect world, you would be tracking all these sources of security info and then perform static analysis of your software to assess the security risk. But the amount of time and resources invested in that would be massive. There is what they call the point of diminishing returns. At a certain point, a vulnerability poses such a low level of risk that it makes no sense to expend any time to investigate and fix it.

The challenge of how to deploy your resources and how to prioritize security issues is even greater when you consider that there are something like 300 new vulnerabilities disclosed in the NVD every week.

So we advise our customers to make security maintenance a part of the product development and product maintenance processes, and central to that effort is monitoring and managing CVEs.

Adam: That's sound advice. Overall, what approaches do you see developers and engineering managers taking to monitoring and addressing CVEs?

Akshay: Well, the easiest approach is to do nothing at all and hope for the best. But that's clearly a non-starter and, very frankly, irresponsible.

A common approach to dealing with CVEs is to monitor, analyze and address them manually, in a Do-It-Yourself (DIY) process.

While that might seem straight-forward, it's really the least effective and most costly approach, when you consider the time and effort it takes to analyze 300 CVEs every week. On top of that, the same team needs to be monitoring patches and software versions of all components across all your products.

And oh, by the way, these same people probably are involved in actually delivering product and enhancements to market. So they need to find some time to work on that too, right?

It's really overwhelming to do the manual approach and expect to have a clear view of vulnerabilities affecting your products because there are so many moving parts.

Adam: What does that mean? What are the moving parts?

Akshay: On the one hand, there are the contents of your product itself. You need to have current information on all packages and versions across all the products in development and in maintenance or support. That of course can change at any time, so you need a way to constantly monitor your own product components.

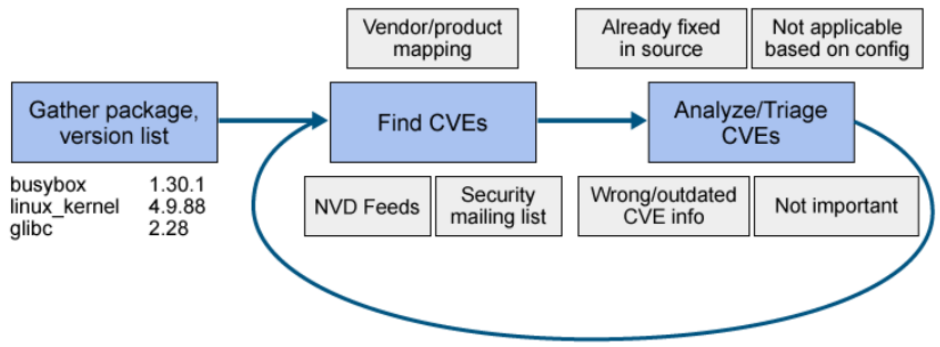
On the other hand, there is the flood of CVEs that you need to review, sift through and match to your product contents.

Then there are the updates, patches, upgrades and enhancements released by third parties who support the components in your product.

"False positives in particular can be a real problem if you are trying to do this all manually. Picture having your developers expending cycles on what seems to be a critical security flaw, only to find out an outdated record created a false positive alert about a vulnerability match with your product. It's critical to avoid wasting time on such wild good chases."

All three of these sources of data need to be monitored and analyzed as shown below.

Do-It-Yourself (DIY) CVE monitoring process



When you arrive at a list of the CVEs affecting your products, you then need to triage the list, analyzing them to determine which are the most pressing to address because they pose the greatest risk for your products based on their configurations and how they are deployed.

All these things need to be accomplished and repeated in a consistent way. And that's before you have actually addressed even a single vulnerability.

Adam: CVE monitoring is built in to Yocto. Doesn't that make the process simpler if you use Yocto?

Akshay: That can help. But the process of pinpointing the specific components that are affected in your builds is still very cumbersome in Yocto's CVE readouts.

And even worse, our analysis of Yocto CVE monitoring shows an excessive number of false positives and CVE misses. That's because much of the NVD data has naming inconsistencies, simple typos in data records, outdated information or missing data.

Any of these can cause Yocto CVE monitoring to report a CVE match where none exists or miss CVEs that should have been a match.

False positives in particular can be a real problem if you are trying to do this all manually. Picture having your developers expending cycles on what seems to be a critical security flaw, only to find out an outdated record created a false positive alert about a vulnerability match with your product. It's critical to avoid wasting time on such wild goose chases.

Adam: So let's say we have assembled a list of CVEs we believe affect our product line. What's next? How do you triage and prioritize them?

Akshay: The step of triaging the CVEs is just like in the hospital or medical care setting. Emergency room doctors will evaluate patients coming in and move those with the most dire, pressing conditions to the top of the list for care.

With CVEs, you analyze all the CVEs in that batch and figure out which ones pose the biggest risk and must be fixed right away. Others might be less pressing, with less likelihood of causing a security problem for a customer. And others might be not important at all and can be safely ignored.

Really this prioritization approach is how you turn security maintenance into a manageable process. Say you want to do weekly security reviews of vulnerabilities, which we have seen as a common best practice. Filtering and prioritizing the CVEs is how you can cut through and make sense of the hundreds of vulnerabilities reported each week. Our security maintenance services include tools to make this analysis and triage simple and fast.

So, for example, when you do this triage analysis, you start with the Common Vulnerability Scoring System, or CVSS. That's the commonly used model for scoring a given CVE in a standard way to arrive at a severity rating.

CVSS ratings may rank Low, Medium, High or Critical. The rating then provides some good initial guidance for your comparison and prioritization of one CVE versus another.

But then you also need to consider a lot of other factors.

For example, does the CVE's attack vector matter in your product's deployment? In other words, for an attacker to exploit the CVE, will they be able to launch an attack against your product using that vector?

A common example is a CVE that uses the network attack vector, such as gaining unauthorized remote access to a device. If your product is not going to be deployed as a networked device, or will be on its own entirely isolated, air-gapped network, then this CVE may not be as important to address immediately, even if it has a Critical CVSS rating. It's still important to address such vulnerabilities, such as ones that could be exploited during a product update that introduces new code. But the triage process enables you to differentiate between exploit risks that are pressing and immediate and those that are less likely to occur.

Similarly the configuration of your system and components may make a given CVE irrelevant because a process or function required for an exploit of the CVE is not activated in your product. For example, some medical devices have been compromised in recent years because of exploits taking advantage of remote software updates as the attack vector. These vulnerabilities do not apply if no remote update function is available, or if it has been disabled, as was the step the device manufacturers took to mitigate the vulnerability.

"Similarly, a reason to prioritize CVEs higher is whether an easily applied patch is already available and known and whether a single CVE fix can address multiple of your products that have components in common. That's low-hanging fruit to fix with minimal effort and so may make sense to take care of quickly."

Another consideration for triaging your list of CVEs is whether a known exploit of a CVE is in the wild.

Sometimes a CVE comes to light because a hacker has already built an exploit to take advantage of a vulnerability that they found on their own. Unfortunately, it might take a damaging security breach in a customer for the CVE to be identified and reported.

But many CVEs are found and reported before that happens because security researchers, bug bounty programs, and responsible disclosure programs are notifying vendors, vulnerability list managers, and security database authorities. That gives product developers a head-start to address CVEs before an exploit appears to take advantage of them.

Similarly, a reason to prioritize CVEs higher is whether an easily applied patch is already available and known and whether a single CVE fix can address multiple of your products that have components in common. That's low-hanging fruit to fix with minimal effort and so may make sense to take care of quickly.

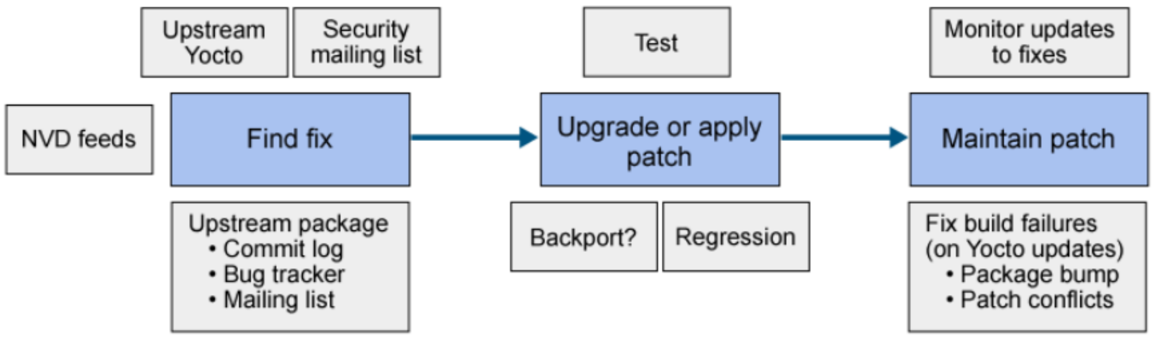
Adam: Along those lines, you mentioned monitoring patches and software upgrades as one of the moving parts to be managed in a security maintenance program. What's the challenge there?

Akshay: Patch management alone is always challenging, especially if you have a large number of open source components. You need to evaluate when to apply a patch, how the patch affects other components, what testing needs to be conducted, whether a patched component can be backported to earlier versions, and so on.

But add to that the pressure of dealing with a high severity CVE, one that might be putting your customers at risk right now, and patch management gets even more challenging.

We guide our customers to follow a consistent process in a CVE-driven patching scenario. Once a given CVE has been identified, your team will investigate the fixes and so may review logs, bug tracker reporters, and other documentation distributed via the mailing lists for upstream packages.

DIY CVE Patching



The right fix will depend on many factors. There might be a patch for a set of issues including the specific CVE. Or it may make more sense to upgrade software versions because a newer version is not affected by the CVE.

We always urge our developer customers to upgrade to the latest version of software components whenever possible. The latest version will address CVEs and possibly other security vulnerabilities that might not be part of the CVE dictionary. This is particularly true for the Linux kernel where upgrading to the latest LTS release is the best bet to keep it secure.

But you also need to consider earlier versions of your code and whether any fixes can be easily backported without breaking product functionality.

The decisions become even more complex if patches or upgrades become available for multiple components in your product at the same time. Analyzing potential conflicts and regression testing become essential steps to manage that type of scenario.

Adam: So is it fair to say that time is the enemy when it comes to monitoring CVEs and addressing them?

Akshay: Yes, time is the enemy in more ways than one.

For example, there often is a gap of days or weeks between the time a vulnerability has been first disclosed, when it has been fully analyzed, and when a patch has been issued.

"Security maintenance and mitigation management workflows should be a central part of all product maintenance reviews and processes. Surprising numbers of companies don't take that active approach but only focus on CVEs when there is a major problem, like a customer with a security breach."

That's your "vulnerability window." It's the period during which the black-hat hacking community has become aware of the vulnerability and is working hard on designing exploits to take advantage of it. If the exploit emerges and spreads before the full analysis is available or the patches are issued, then you may be fully vulnerable to a damaging attack.

Adam: Given all this complexity, what's your advice to engineering managers looking to boil all this down into a simple, manageable process?

Akshay: The first step is recognizing that you need that process at all.

Security maintenance and mitigation management workflows should be a central part of all product maintenance reviews and processes. Surprising numbers of companies don't take that active approach but only focus on CVEs when there is a major problem, like a customer with a security breach.

Once you have established it, the process itself should feature a few key elements.

First, you need tools for monitoring and filtering CVEs based on your actual, accurate product components. As I mentioned earlier, you need a monitoring process less prone to false positives and CVE misses than the Yocto CVE monitoring feature.

Then you need shared workspace and communication tools to support CVE investigation, triage, prioritization and collaboration. That should include the ability for teams to share findings, status and actions, and especially to learn from one another as the process matures. Your process also should include reports and dashboards that enable you to see the big picture and understand the CVE counts by product, the progress in fixing them, and the changes over time.

Then, you need to automate the patch monitoring and management process as much as possible, to accelerate the mitigation of vulnerabilities. That involves automatically identifying patches and upgrades that pertain to the specific product components and enabling your team to assess them as a fix for a particular CVE.

At Timesys we have built these capabilities into Vigiles, our vulnerability monitoring and management service.

Your team can try Vigiles for free here: **https://www.timesys.com/security/vigiles/**

About the interview participants

Akshay Bhat, Director of Engineering, Security Solutions, has experience with embedded systems that spans a broad range of industries with a focus on board bring-up, driver development and software security. Akshay received his MS in Electrical Engineering from NYU Polytechnic University.

Adam Boone, Vice President of Marketing, has 25 years of experience in marketing, communications, strategy, and media, including networking, security and telecom companies. He led marketing for two Sequoia Capital start-ups with successful exits. Adam has worked with dozens of start-ups and early stage companies to engineer growth. He holds an MBA from Arizona State University and completed the Competitive Marketing Strategy Program at University of Pennsylvania's Wharton School.

03

Process Template: Managing Security Maintenance in the Product Lifecycle

Too often it seems the first notification of a software vulnerability comes from an affected customer or the negative publicity surrounding a high-profile data breach.

Then follows the mad scramble to mitigate the vulnerability, notify customers, update products in the field and so on.

This reactive approach to vulnerability management for your embedded system products simply doesn't fly in today's heightened vulnerability environment.

Instead, proactive product security maintenance is today's industry best practice. Companies that develop and maintain embedded system products are increasingly making security maintenance a key focus for product maintenance and enhancement processes.

Proactive security maintenance and vulnerability management boil down to questions of risk management.

Is it better to face the risk of a catastrophic security failure in one of your customers, with all the accompanying damage that entails?

Or is it better to identify vulnerabilities as soon as they emerge and give your team a running start toward addressing them, distributing fixes, and protecting your customers?

Open Source Best Practice: Proactive Security

Open source systems such as embedded Linux and other open source components are increasingly common in embedded systems. The benefits of faster time-to-market and fewer development cycles for broader functionality are undeniable.

But open source also brings with it a basic truth: You are incorporating something into your product that you did not make.

And as with any component acquired from a third party and integrated into your product, you must be diligent about the component's impact on your device's security posture.

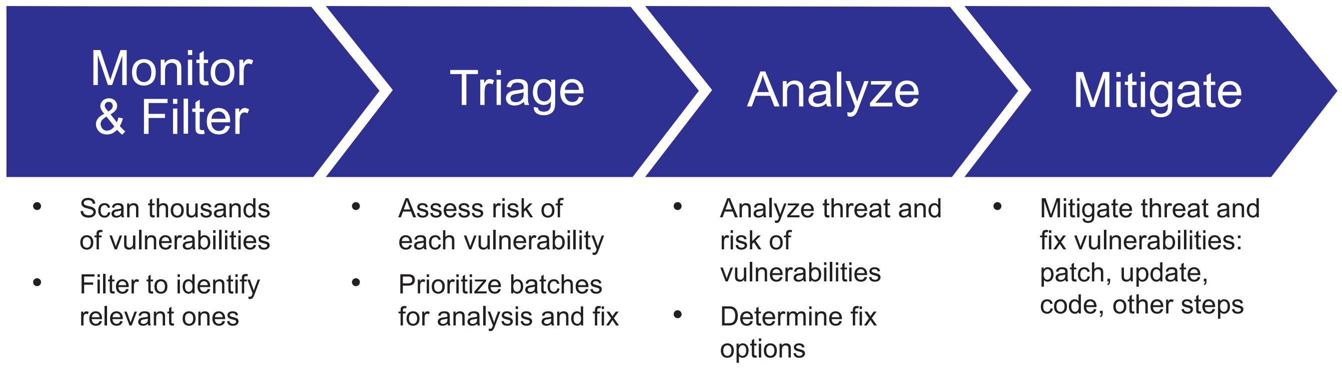
This boils down to the need to continuously monitor for vulnerabilities that may affect that component and create an attack vector that can be exploited by an attacker.

Proactive security maintenance and vulnerability monitoring must be central parts of the product maintenance and enhancement process.

Process Template: Security Maintenance Steps

Security maintenance and vulnerability monitoring for embedded products can be broken into four primary steps that can be integrated into your product maintenance processes throughout your product lifecycle. Each step looks to answer several questions or address certain tasks that will enable your security maintenance process to proactively address vulnerabilities in an efficient way.

Security Maintenance Steps

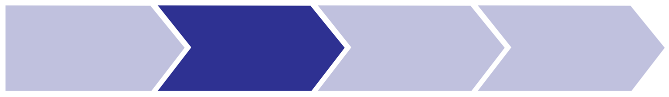

Monitor & Filter
Triage
Analyze
Mitigate
• Scan thousands of vulnerabilities
• Filter to identify relevant ones
• Assess risk of each vulnerability
• Prioritize batches for analysis and fix
• Analyze threat and risk of vulnerabilities
• Determine fix options
• Mitigate threat and fix vulnerabilities: patch, update, code, other steps

1. Vulnerability Monitoring & Filtering

Tasks at the monitoring stage of security maintenance look to answer questions like:

- Do newly reported vulnerabilities affect our products in development or in maintenance?
- Which specific versions of our product software or third-party components in our software are affected by a given vulnerability?
- Are older vulnerabilities now relevant to our products because of a change in the product, such as a component software update or how it is deployed?

Our Vigiles vulnerability monitoring and management service enables your team to easily filter out irrelevant vulnerabilities and false positives, permitting them to quickly focus on the vulnerabilities that really matter.



2. Vulnerability Triage & Prioritization

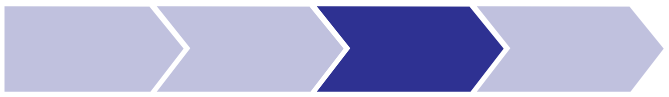
Once the relevant vulnerabilities in a given period of your product maintenance cycle have been identified, the next step is to assess the severity of them for your specific use cases and then prioritize fixing them.

So, for example, a particular vulnerability may be given a high severity score under the Common Vulnerability Scoring System (CVSS) that is widely used for assessing vulnerability risk. But the attack vector of that vulnerability may not be exposed in your product, meaning its severity for your particular configuration is low.

Questions to answer at the Triage & Prioritization stage include:

- Which of the identified vulnerabilities are the highest risk for our customers and products, given product configurations, deployment modes, number of units in production deployment, and exposed attack vectors?
- Which vulnerabilities affect multiple of our products, or may be common across many versions of our products because of replicated components?
- Which vulnerabilities are believed to be already addressed by a patch or update from us or an upstream supplier?

Which vulnerabilities appear to require new development or enhancements to fix? Can multiple vulnerabilities or multiple product lines potentially be addressed by a common enhancement or update or patch?



3. Analysis & Remediation Planning

Communication and a clear division of tasks are key for efficient analysis and mitigation of individual vulnerabilities. This requires tools that support collaboration and easy documentation sharing among users and across teams.

This type of collaborative analysis and planning is especially important if it is believed that a common patch may apply to fix a vulnerability across multiple product lines. Teams operating in silos may waste cycles on redundant mitigation steps.

Questions being addressed at this stage include:

- Which known updates or patches for in-house or third-party software will address the vulnerability?
- Does the vulnerability require immediate steps to be taken by customers to reduce risk of an exploit or security breach? What are those steps and how should notification be conducted?
- If a vulnerability requires development work to fix, when can it be addressed in the development cycle and how will that impact other projects, testing, and so on?
- If a vulnerability demands generating and distributing a patch immediately, what are the require-ments and plan for development and distribution?

Our Vigiles vulnerability monitoring and management service enables your team to easily collaborate and share information about vulnerabilities, available fixes, and investigation findings.

4. Mitigation

The final step in an effective security maintenance process is the mitigation of the identified vulnerabilities.

Some mitigation steps may be temporary measures, such as notifying customers and requiring them to take a unit offline or change its configuration until a patch can be issued.

Other mitigation steps will be more involved, requiring development work, incorporating new versions of components into the system, testing, issuing updates, and so on.

Our Vigiles vulnerability monitoring and management service can automatically identify suggested fixes for vulnerabilities by matching vulnerabilities with versions or patches that address them and flagging these fixes for your team.

Learn more at: **https://www.timesys.com/security/vigiles/**

04

Timesys Vigiles: Vulnerability Management & Mitigation for Stronger Embedded System Security

A giant list of vulnerabilities does little to help you ensure your products are secure.

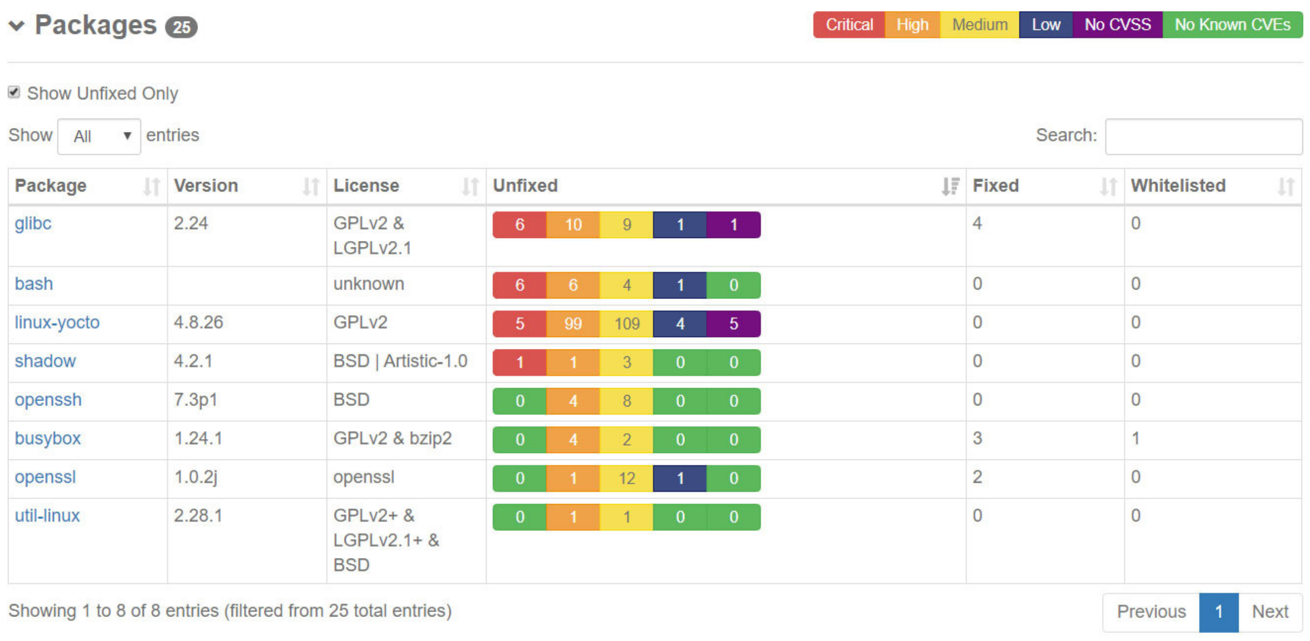
What matters is how you filter the list, triage the vulnerabilities, and mitigate the ones that pose the greatest risk.

In addition, Gartner reports that attackers are increasingly targeting open source components, essentially getting a foothold earlier in the software supply chain. This means that downstream product makers who incorporate open source components into their products are exposing their products and customers to potential compromise and higher risk.

The Timesys Vigiles Security Monitoring & Management Service enables embedded system developers to maintain stronger product security throughout their product lifecycles. Vigiles automates, simplifies and accelerates vulnerability monitoring, exposure assessment and mitigation. Features and capabilities include:

Vulnerability Management & Mitigation Optimized for Embedded

Vigiles integrates with embedded system software development tools including Yocto, Buildroot and Timesys Factory, along with other SDLC and CI/CD tools using APIs. The service provides end-to-end workflow support for developer-driven vulnerability tracking, investigation and mitigation.



Packages 25

Critical | High | Medium | Low | No CVSS | No Known CVEs

☑ Show Unfixed Only

Show All entries

Search:

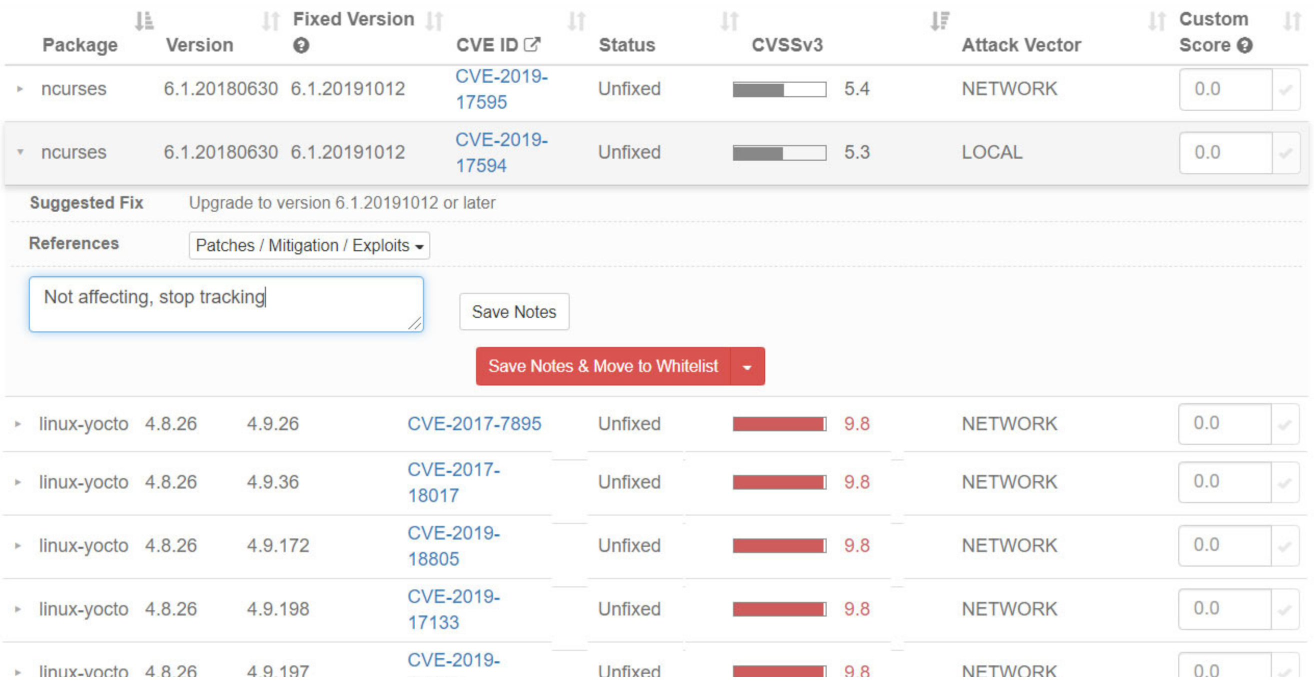
Package	Version	License	Unfixed	Fixed	Whitelisted
glibc	2.24	GPLv2 & LGPLv2.1	6 10 9 1 1	4	0
bash		unknown	6 6 4 1 0	0	0
linux-yocto	4.8.26	GPLv2	5 99 109 4 5	0	0
shadow	4.2.1	BSD | Artistic-1.0	1 1 3 0 0	0	0
openssh	7.3p1	BSD	0 4 8 0 0	0	0
busybox	1.24.1	GPLv2 & bzip2	0 4 2 0 0	3	1
openssl	1.0.2j	openssl	0 1 12 1 0	2	0
util-linux	2.28.1	GPLv2+ & LGPLv2.1+ & BSD	0 1 1 0 0	0	0

Showing 1 to 8 of 8 entries (filtered from 25 total entries)

Previous 1 Next

Vigiles is optimized for embedded systems and filters vulnerabilities based on a project’s Linux kernel configuration and U-Boot configuration, which removes vulnerabilities affecting features not being used. This reduces vulnerability investigation and triage tasks by 75 percent on average.

The service reports only those vulnerabilities affecting packages installed on your target. It automatically tracks CVEs already fixed in your Yocto/Buildroot implementation and Includes support for reporting CPU/SoC vulnerabilities.



Package	Version	Fixed Version	CVE ID	Status	CVSSv3	Attack Vector	Custom Score
ncurses	6.1.20180630	6.1.20191012	CVE-2019-17595	Unfixed	5.4	NETWORK	0.0
ncurses	6.1.20180630	6.1.20191012	CVE-2019-17594	Unfixed	5.3	LOCAL	0.0

Suggested Fix Upgrade to version 6.1.20191012 or later

References Patches / Mitigation / Exploits

Not affecting, stop tracking

Package	Version	Fixed Version	CVE ID	Status	CVSSv3	Attack Vector	Custom Score
linux-yocto	4.8.26	4.9.26	CVE-2017-7895	Unfixed	9.8	NETWORK	0.0
linux-yocto	4.8.26	4.9.36	CVE-2017-18017	Unfixed	9.8	NETWORK	0.0
linux-yocto	4.8.26	4.9.172	CVE-2019-18805	Unfixed	9.8	NETWORK	0.0
linux-yocto	4.8.26	4.9.198	CVE-2019-17133	Unfixed	9.8	NETWORK	0.0
linux-yocto	4.8.26	4.9.197	CVE-2019-	Unfixed	9.8	NETWORK	0.0

Vulnerability mitigation is expedited because Vigiles automatically identifies “suggested fixes” such as patches or updates of components that will mitigate vulnerabilities.

Flexible, intuitive team collaboration and sharing tools ensure your mitigation efforts are efficient and fast. Powerful and convenient “comparison” capabilities give you quick insight into what has changed between builds.

Software Composition Analysis (SCA) for Embedded

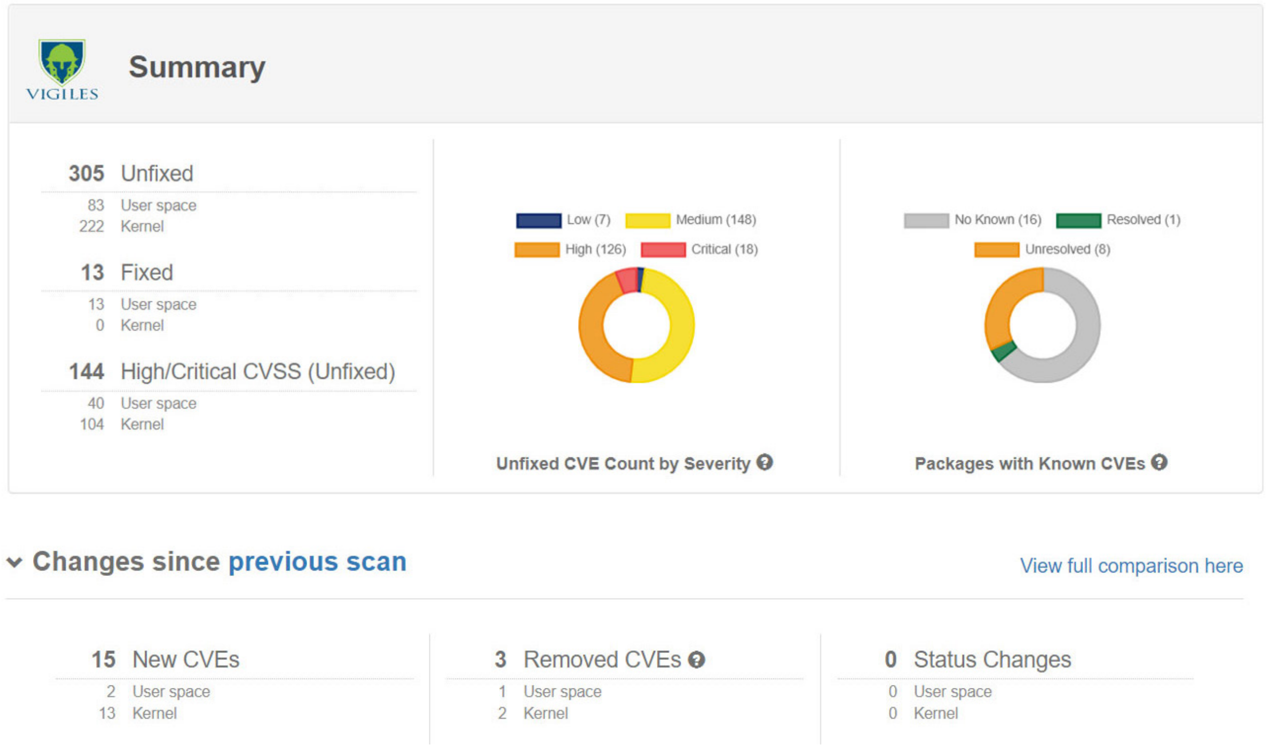
Vigiles’ SCA functionality will automatically generate a Software Bill of Materials (SBOM) for Yocto, Buildroot and Timesys Factory projects.

Now you can understand exactly which open source third-party components are in your products and which vulnerabilities pertain to them. Features include detailed vulnerability reports, trend reports, summaries and a searchable vulnerability database.

This significantly reduces the level of effort to sift through and analyze vulnerabilities because your team can focus on only those that matter.

Superior Vulnerability Data

Vigiles delivers superior, highly accurate vulnerability data, augmenting the feed from the NVD with multiple additional vulnerability feeds. The Timesys security team curates vulnerability data, which reduces false positives and produces a 40 percent improvement in data accuracy compared to the NVD.



Summary

305 Unfixed
83 User space
222 Kernel

13 Fixed
13 User space
0 Kernel

144 High/Critical CVSS (Unfixed)
40 User space
104 Kernel

Changes since previous scan

View full comparison here

15 New CVEs
2 User space
13 Kernel

3 Removed CVEs
1 User space
2 Kernel

0 Status Changes
0 User space
0 Kernel

You can receive expedited notification of newly reported vulnerabilities as much as four weeks earlier than from the NVD.

Simple One-Stop-Shop Solution

Vigiles encompasses vulnerability monitoring, triage and mitigation in one easy-to-use solution. Intuitive prioritization and filtering mechanisms get your team going without extensive training or configuration hassles.

The complete vulnerability management workflow is at your fingertips, including history, reports, notes, vulnerability whitelisting and more.

Free Version Available

Vigiles is available in three versions including a free service providing basic vulnerability monitoring. Learn more: **https://www.timesys.com/security/vigiles-vulnerability-monitoring-patch-notification/**