Lifecycle Maintenance of Your BSP

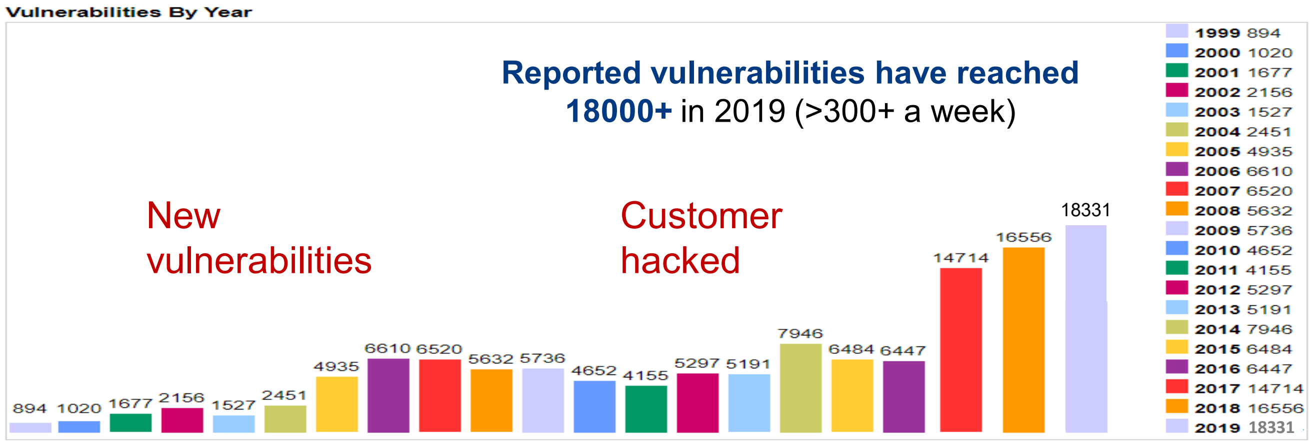
Let us handle the periodic updates for you!

NXP Webinar: June 2, 2020

Presented by: Maciej Halasz

SECURE CONNECTIONS
FOR A SMARTER WORLD

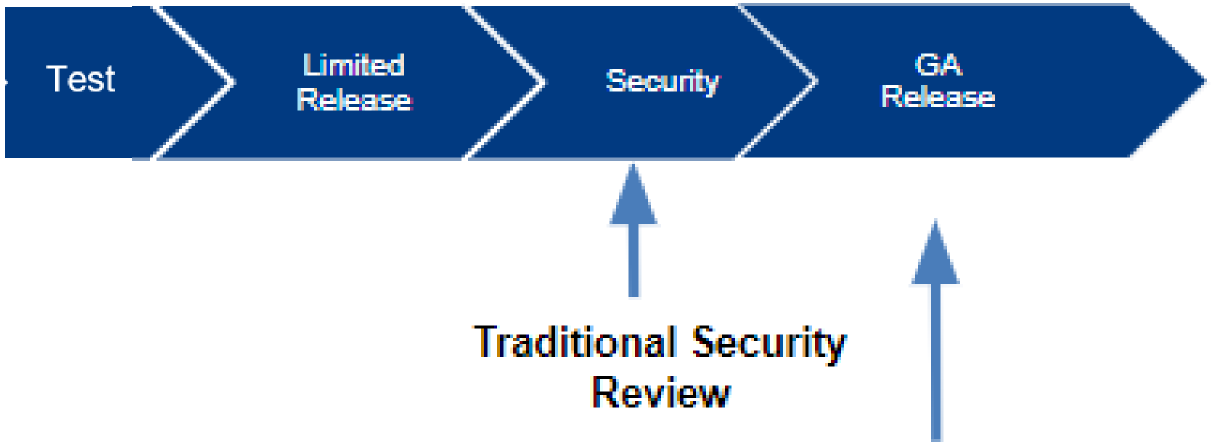
Problem 1: The World is not Frozen, Even if Your Software Is



Reported vulnerabilities have reached 18000+ in 2019 (>300+ a week)

New vulnerabilities

Customer hacked



Traditional Security Review

Embedded Software Frozen Until New Generation

New compliance / security rules

Frequent kernel updates

New deployment modes (connected devices, IoT)

New 3rd party component versions

External Changes

Team is focused on new products

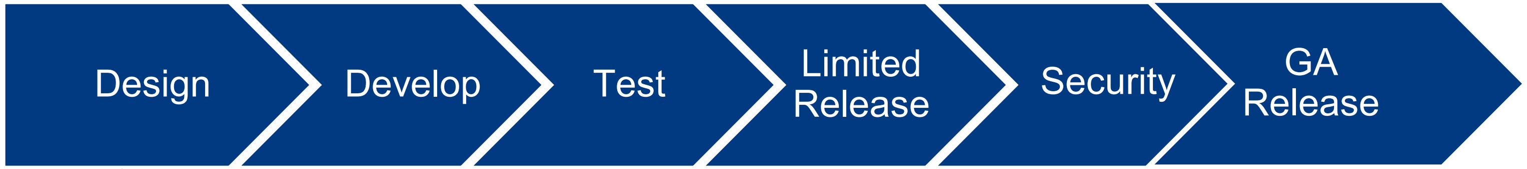
Difficulty analyzing flood of CVEs

No cycles for retesting

Backlog of patches and updates

Internal Challenges

Problem 2: Market Security Requirements are Critical to Customer Acceptance



Design

Develop

Test

Limited Release

Security

GA Release

FDA Guidelines

HIPAA privacy

SCADA security requirements

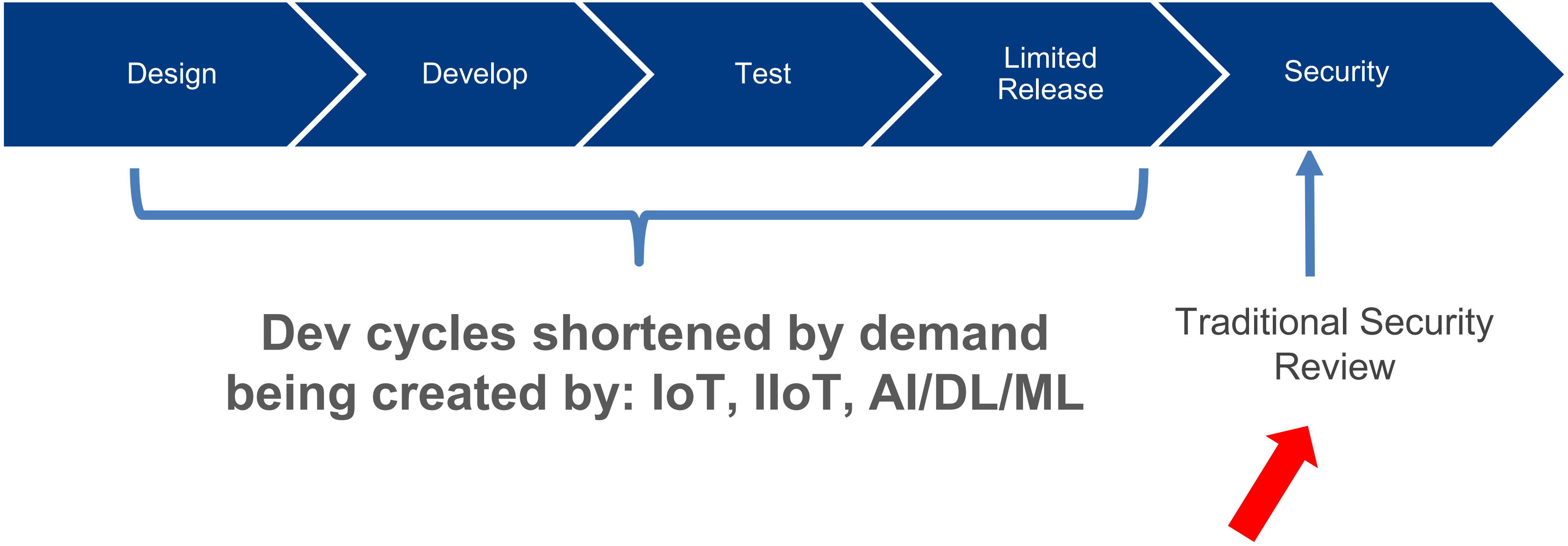
IEC 62304

ICS, IIoT security requirements

End customer security requirements are Growing more complex and are Critical to customer acceptance

Must be baked into product from start

Problem 3: Shorten Development Cycle with Predictable Schedules

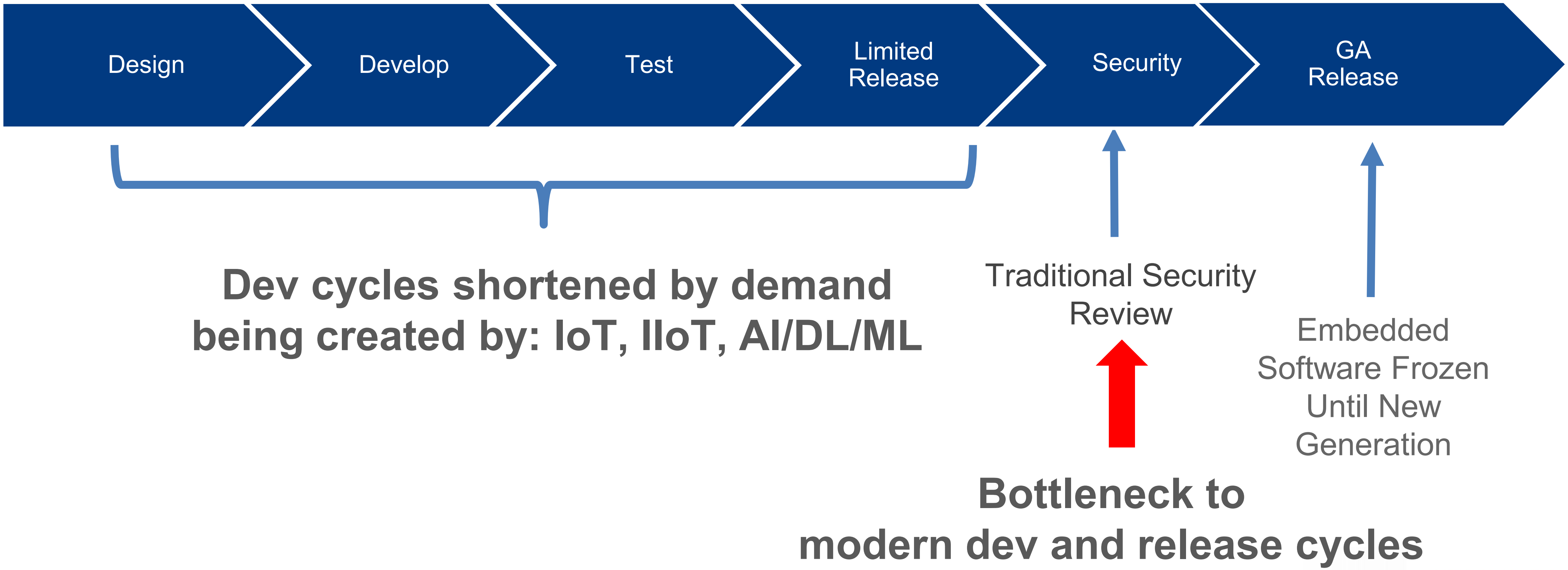


Dev cycles shortened by demand being created by: IoT, IIoT, AI/DL/ML

Traditional Security Review

Bottleneck to modern dev and release cycles

Problem 4: No Longer Ignore Software in the Field



Design

Develop

Test

Limited Release

Security

GA Release

Dev cycles shortened by demand being created by: IoT, IIoT, AI/DL/ML

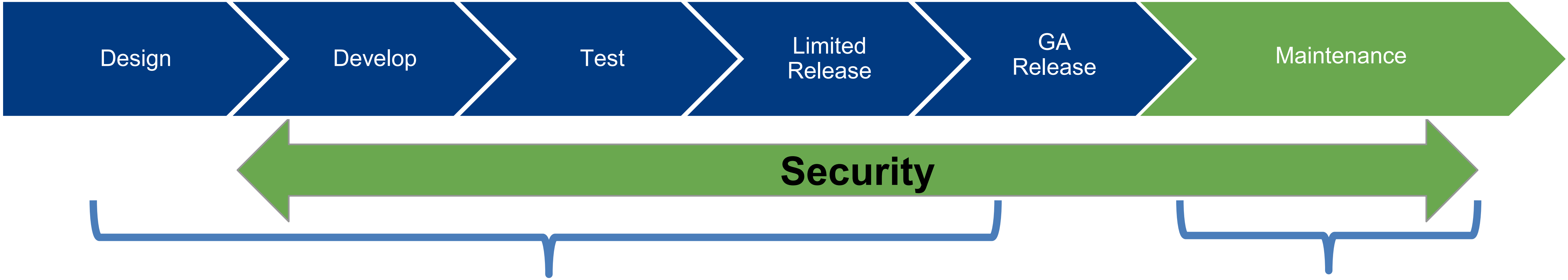
Traditional Security Review

Embedded Software Frozen Until New Generation

Bottleneck to modern dev and release cycles

Solution: Shift Security *Left and Stretch Right*

Active, Continuous Security at Every Stage of SDLC



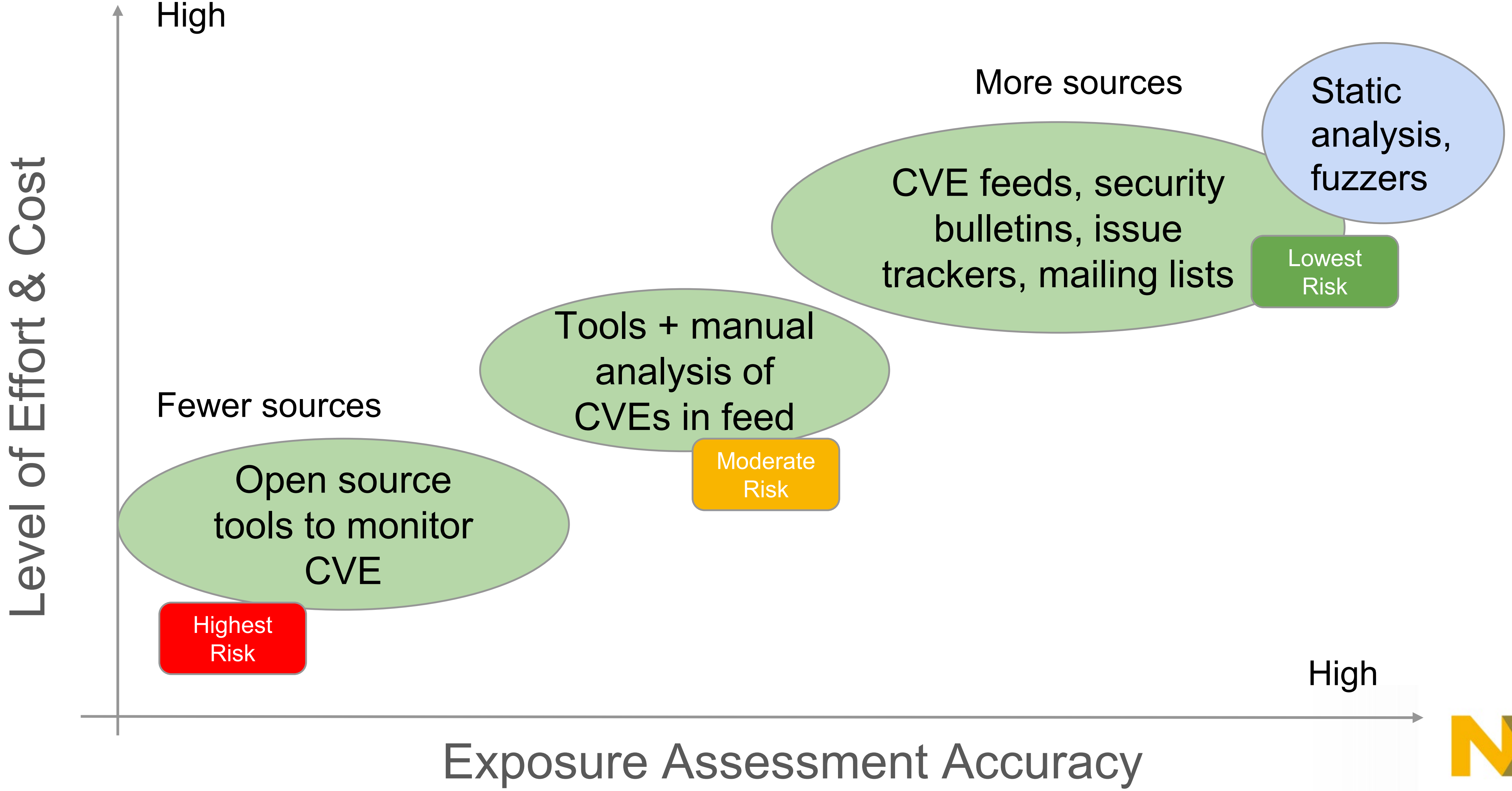
Security in design, development, testing

- Need security tools that are aligned with development workflows and tools
- Need highly accurate vulnerability identification for all versions, all components, all branches
- Need to build using latest, most secure third party components

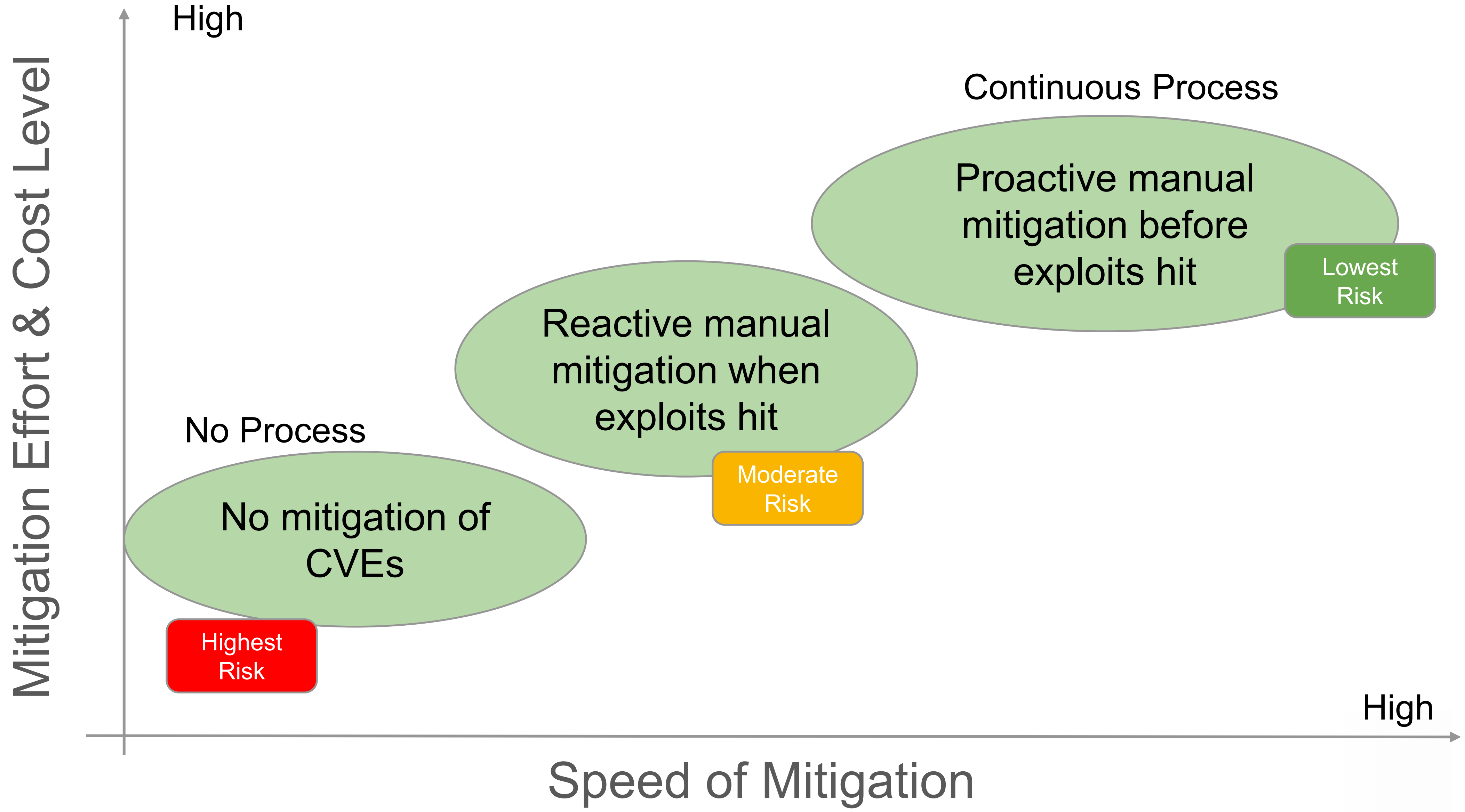
Ongoing developer-driven security maintenance

- Must conduct continuous vulnerability monitoring, patching, and software updates to keep devices secure
- Testing a bottleneck for many
- Accurate vulnerability data and fewer false positives to minimize dev team impacts

Exposure Assessment Effort & Cost


Level of Effort & Cost
High
More sources
Static analysis, fuzzers
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 CVE
Highest Risk
High
Exposure Assessment Accuracy

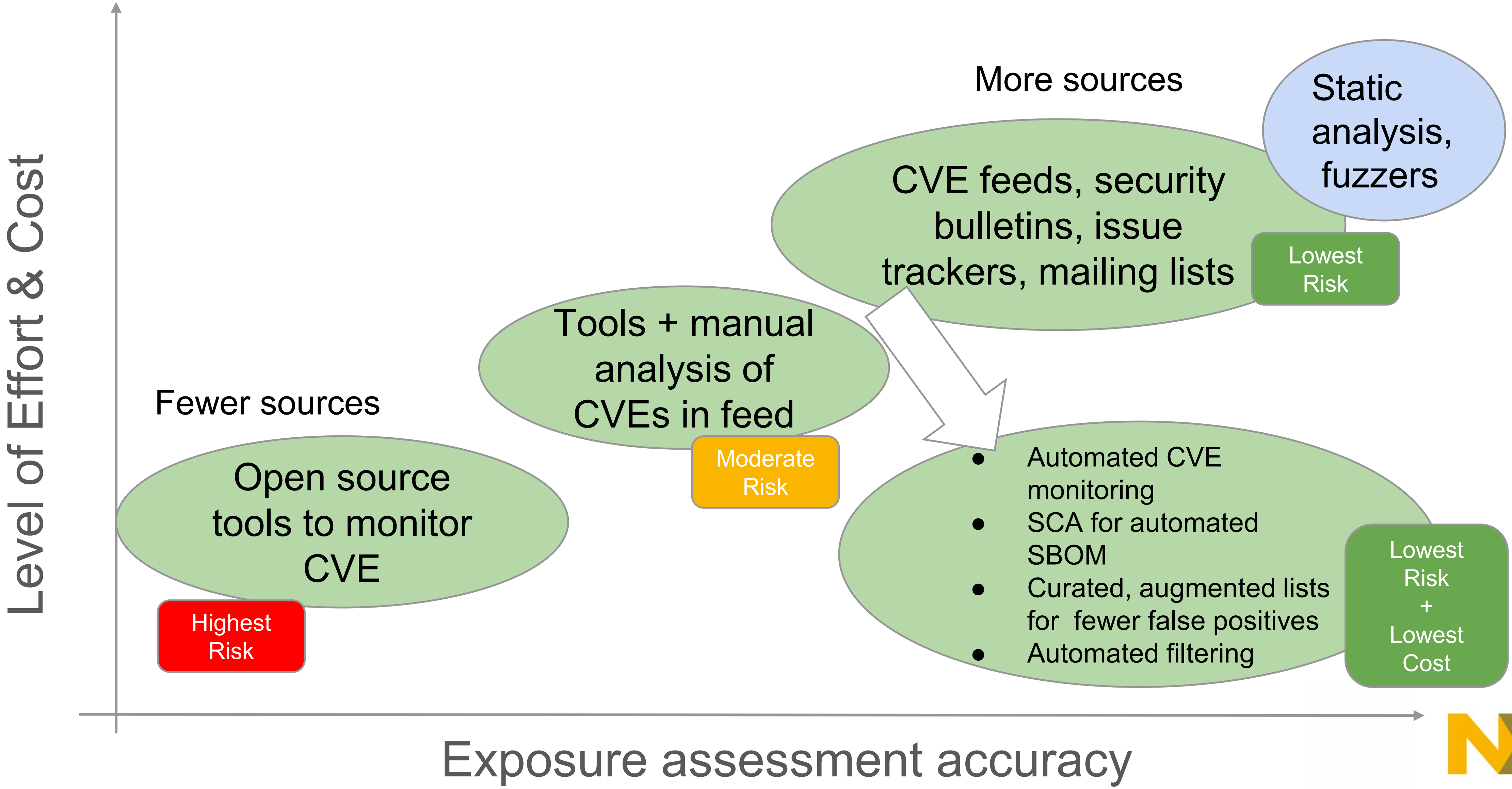
Mitigation Effort & Cost


High
Mitigation Effort & Cost Level
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
High
Speed of Mitigation

How Can You “Jump the Curve”?

- Automated software analysis & SBOM generation
- Automated & augmented feeds & filtering
- Collaboration & sharing across teams
- Automation-assisted analysis & mitigation steps
- Choose tools that are optimized for your particular product areas

Jump the Curve: Exposure Assessment



Level of Effort & Cost

Fewer sources

Open source tools to monitor CVE

Highest Risk

Tools + manual analysis of CVEs in feed

Moderate Risk

More sources

CVE feeds, security bulletins, issue trackers, mailing lists

Static analysis, fuzzers

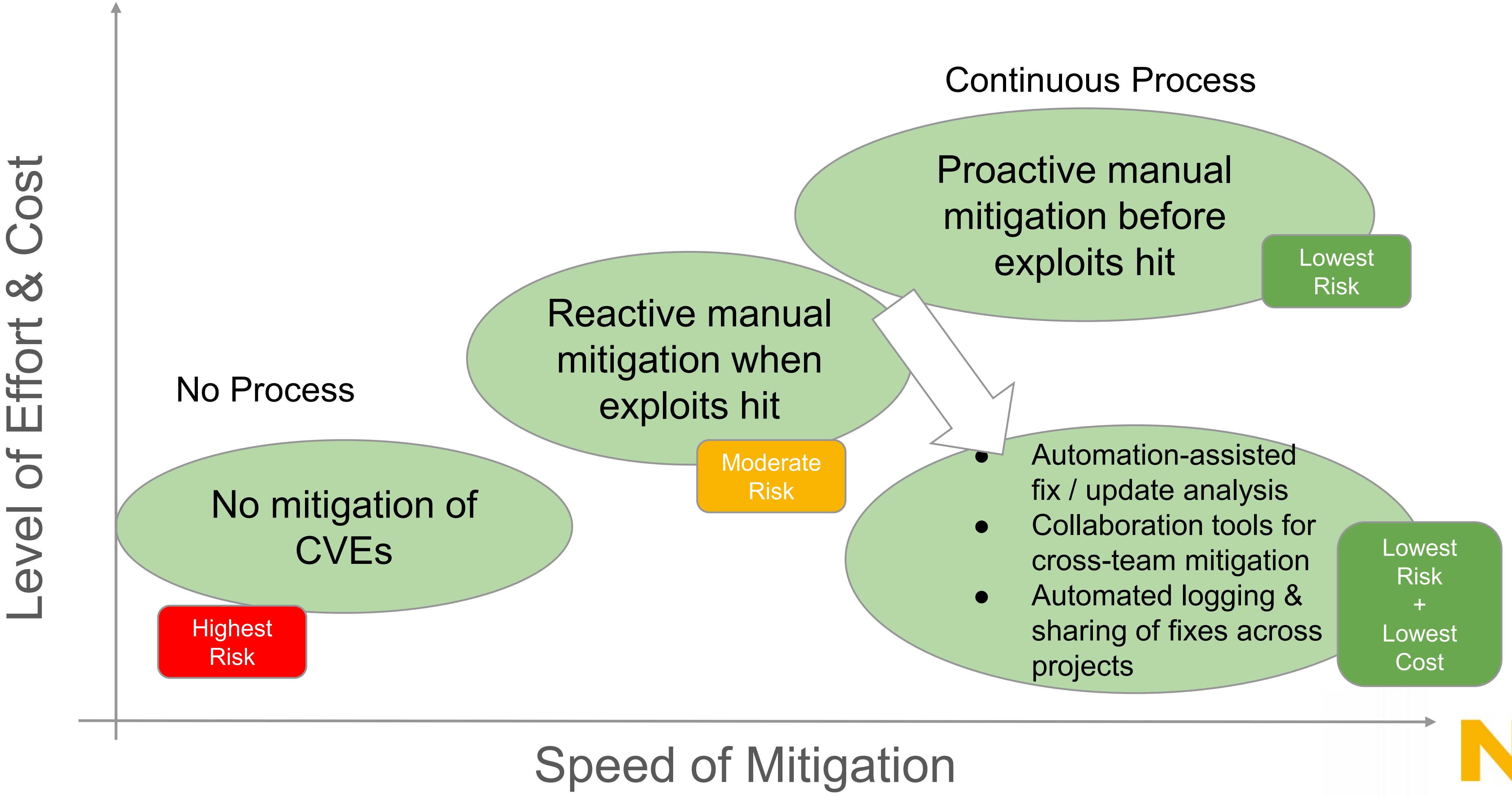
Lowest Risk

- Automated CVE monitoring
- SCA for automated SBOM
- Curated, augmented lists for fewer false positives
- Automated filtering

Lowest Risk + Lowest Cost

Exposure assessment accuracy

Jump the Curve: Mitigation



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

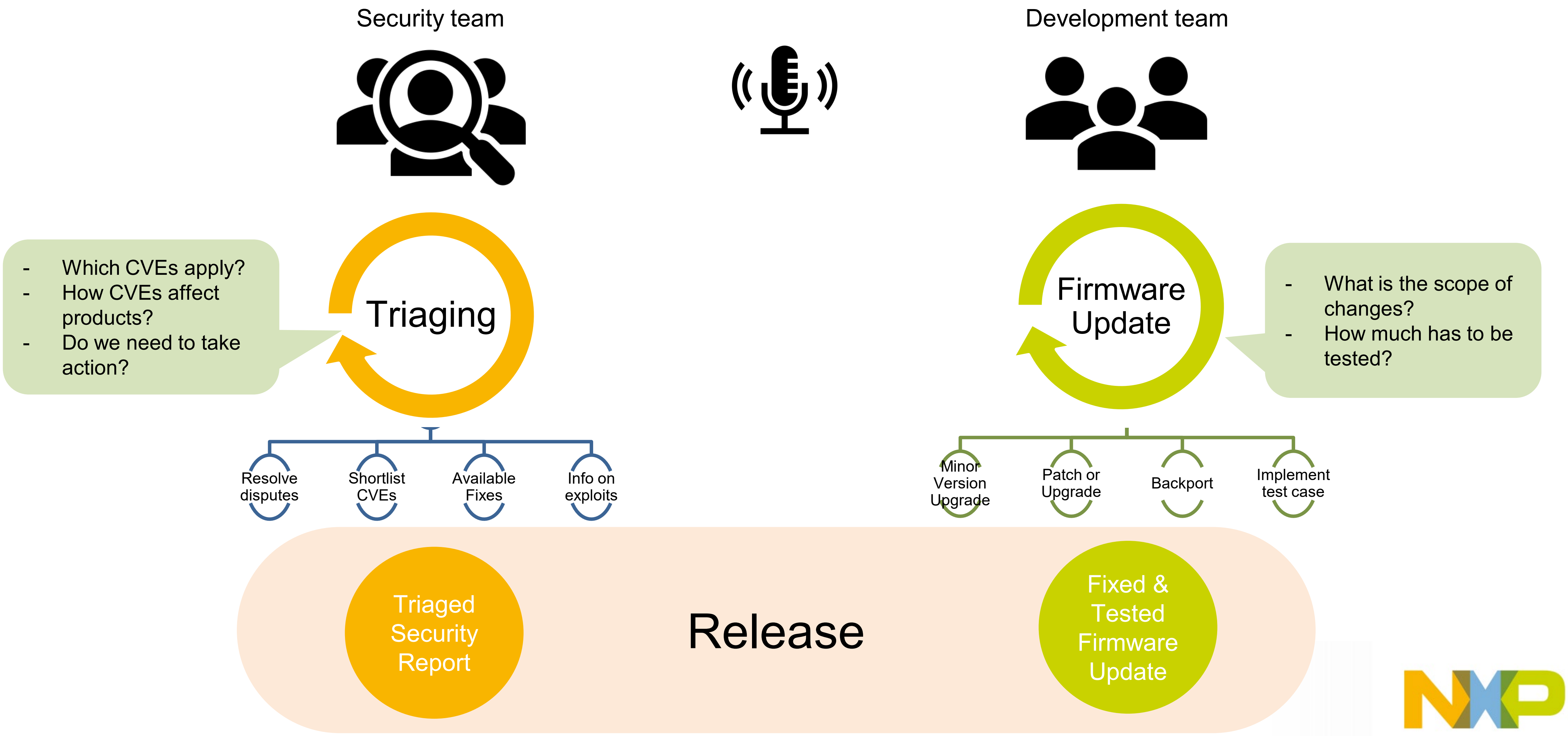
Lowest Risk + Lowest Cost

Security Monitoring Tools

Why monitoring tools are useful?

- Improved security
  - More coverage, better accuracy, early notification
- Time saved in monitoring
  - Identifies/notifies on newly discovered CVEs and fixes
- Reduced triage burden
  - Advanced filtering, fewer false positives, identifies already fixed CVEs
- Workflow management
  - History, collaboration tools, notes, whitelist, exported reports
- Integrates into engineering process
  - Plugs into Yocto, and a vulnerability scan can be triggered for every build
- Simplified, efficient vulnerability maintenance & continuous monitoring
  - Filters CVEs to only those that matter, tools for rapid investigation and mitigation

BSP Maintenance Process



Security team

- Which CVEs apply?
- How CVEs affect products?
- Do we need to take action?

Triaging

Resolve disputes

Shortlist CVEs

Available Fixes

Info on exploits

Development team

Firmware Update

- What is the scope of changes?
- How much has to be tested?

Minor Version Upgrade

Patch or Upgrade

Backport

Implement test case

Triaged Security Report

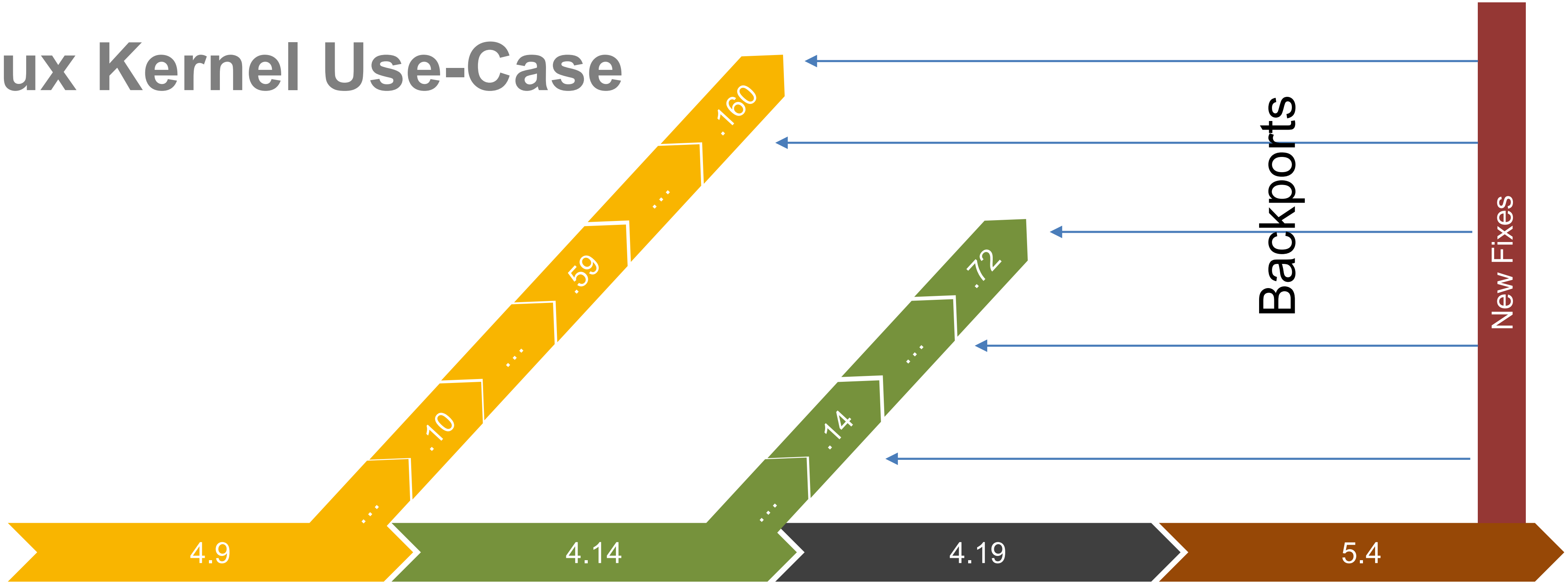
Release

Fixed & Tested Firmware Update

Upgrade or Patch or Backport?

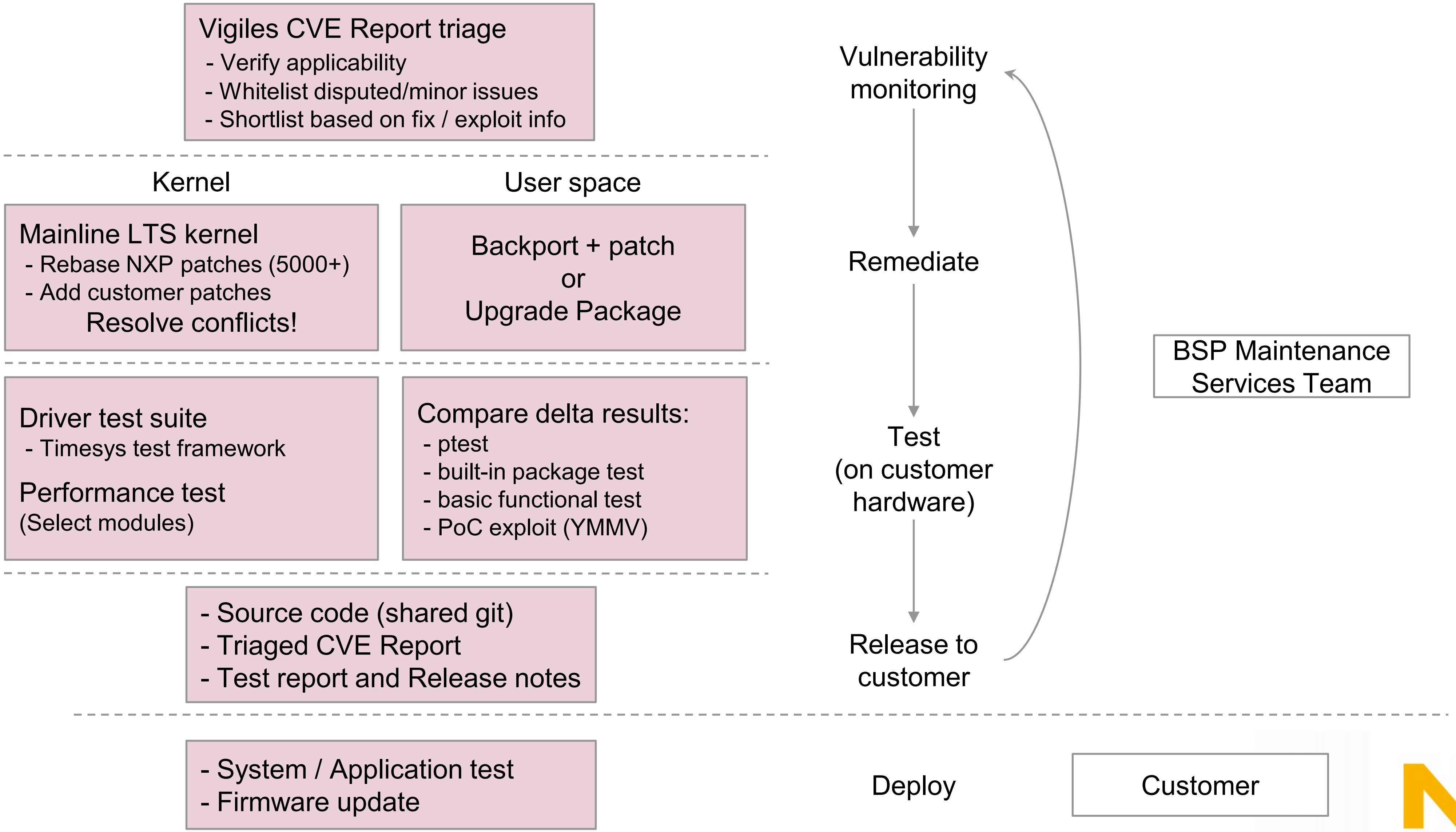
- When to Upgrade
  - Fix implemented in a newer version
  - No License change
  - Understood/minimal/contained impact on other software
- When to Patch
  - Minimize the scope of changes
  - Patch available but new version not released
  - New software version also changes API (backport)
    - API changes risk impacting other softwares resulting in instability
  - Locked/certified software versions
- When to Remove
  - Issues unfixed upstream (abandoned)
  - Unacceptable license change in new version

Linux Kernel Use-Case



- CVE fixes are backported by LTS maintainers
- Minor kernel updates are limited in scope of changes
- Minor kernel upgrades come before custom patches! – Need to adjust!
- Major kernel upgrade may be required when LTS version goes out of maintenance

BSP Maintenance Workflow: *How we do it*



Vigiles CVE Report triage
- Verify applicability
- Whitelist disputed/minor issues
- Shortlist based on fix / exploit info

Kernel

Mainline LTS kernel
- Rebase NXP patches (5000+)
- Add customer patches

Resolve conflicts!

User space

Backport + patch
or
Upgrade Package

Driver test suite
- Timesys test framework

Performance test
(Select modules)

Compare delta results:
- ptest
- built-in package test
- basic functional test
- PoC exploit (YMMV)

- Source code (shared git)
- Triaged CVE Report
- Test report and Release notes

- System / Application test
- Firmware update

Vulnerability monitoring

Remediate

Test (on customer hardware)

Release to customer

BSP Maintenance Services Team

Deploy

Customer

BSP Maintenance Tasks and Staffing Considerations: Stretch Right

Vulnerability monitoring

- Requires dedicated team to filter, analyze, triage, remediate
- Analyze applicability and impact of the vulnerabilities

Kernel updates

- Linux engineering resources to keep up with LTS branch & kernel patches and minor versions

Toolchain updates

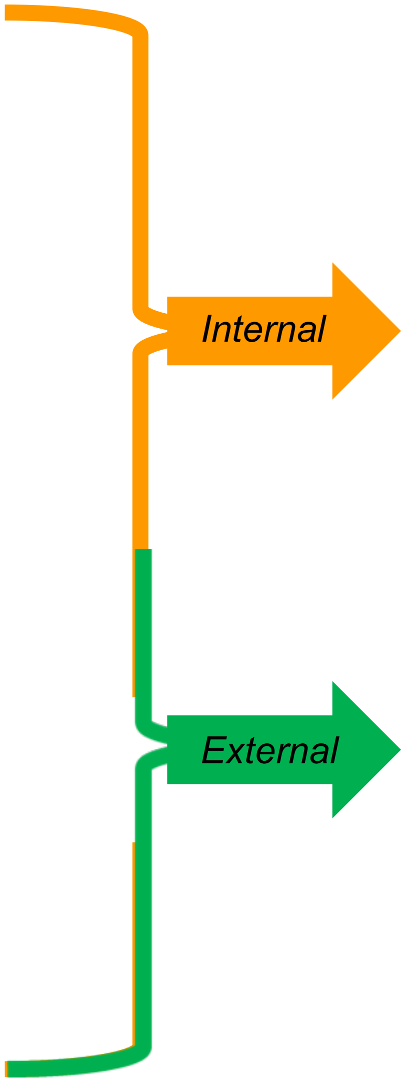
- Toolchain engineering for gcc, glibc bug fixes, security patches
- Pin tool chain version to specific build system (e.g. Yocto)
- Rebuild SDK for application, regression testing

BSP updates

- BSP engineering for updates to libraries and packages (Root File System)
- Integrate and Test patches/updates

Testing and re-testing

- QA Engineers for re-testing of Linux BSP/platform, functional testing of drivers



Could you do all this with a single resource?
How about two resources?
How about a dedicated team of resources?

Frequent maintenance cycles, high staffing costs, priority conflicts

With tight development budgets and product schedules, this work typically gets sacrificed by R&D.

Offload to a turnkey BSP maintenance service

What if you could do ALL this with less than half the cost of a junior engineer?

No brainer, right?

The Hidden Costs of BSP Maintenance

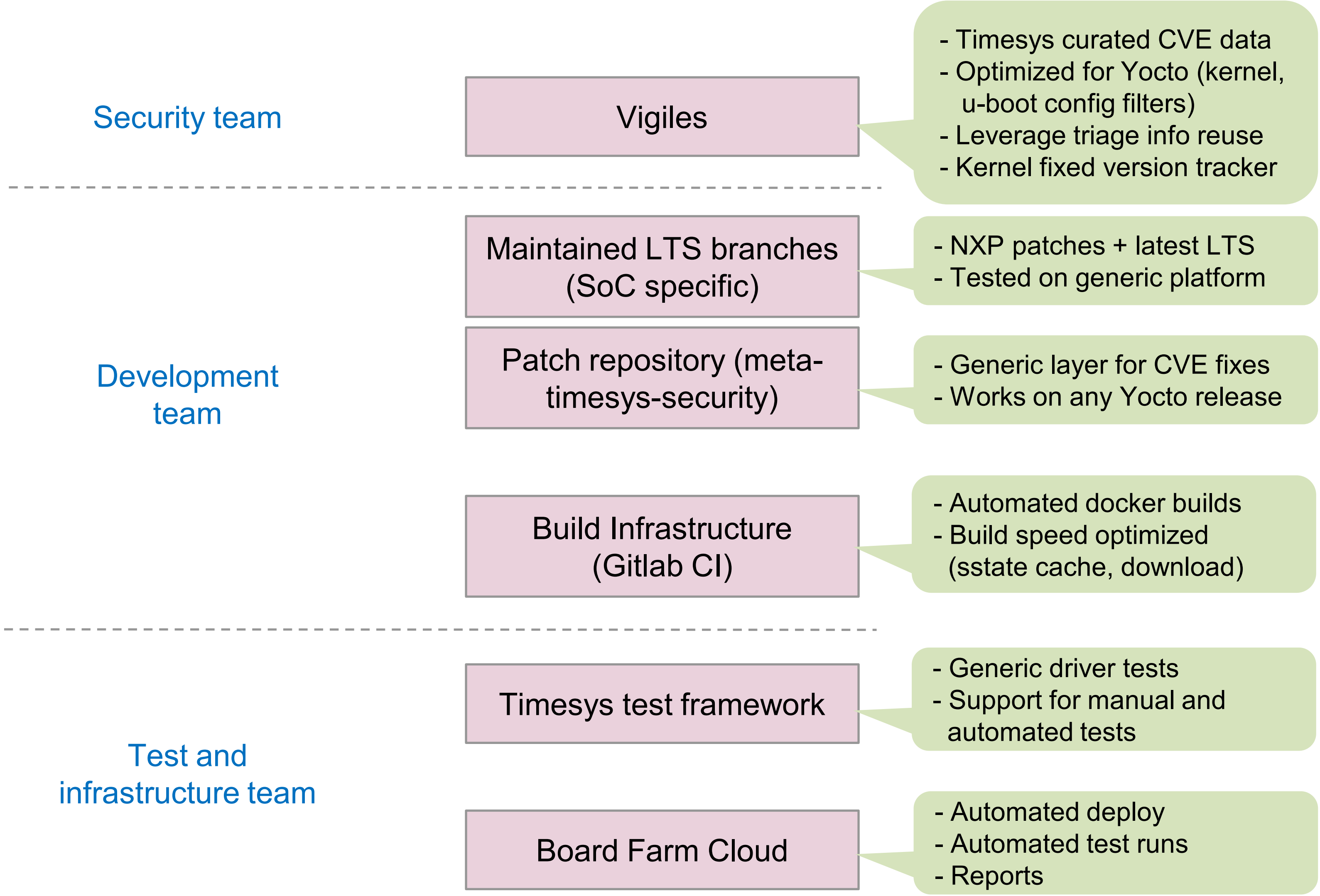
Tasks	1st Board	3 Boards*	5 Boards*
Monitoring	$20k	$25k	$30k
Finding & Applying Patches Finding Fixed Versions & Upgrading Versions	$38k	$50k	$60k
Testing 2 Releases Per Year	$32k	$75k	$120k
Total	**$90k**	**$150k**	**$215k**

BSP Maintenance

Do It Yourself: **$150,000 / year**

**Assume more than 75% overlap in Software components and kernel configurations*

Automation, Scale & Cost Reduction: *How we do it*



Security team

Vigiles

- Timesys curated CVE data
- Optimized for Yocto (kernel, u-boot config filters)
- Leverage triage info reuse
- Kernel fixed version tracker

Development team

Maintained LTS branches (SoC specific)

- NXP patches + latest LTS
- Tested on generic platform

Patch repository (meta-timesys-security)

- Generic layer for CVE fixes
- Works on any Yocto release

Build Infrastructure (Gitlab CI)

- Automated docker builds
- Build speed optimized (sstate cache, download)

Test and infrastructure team

Timesys test framework

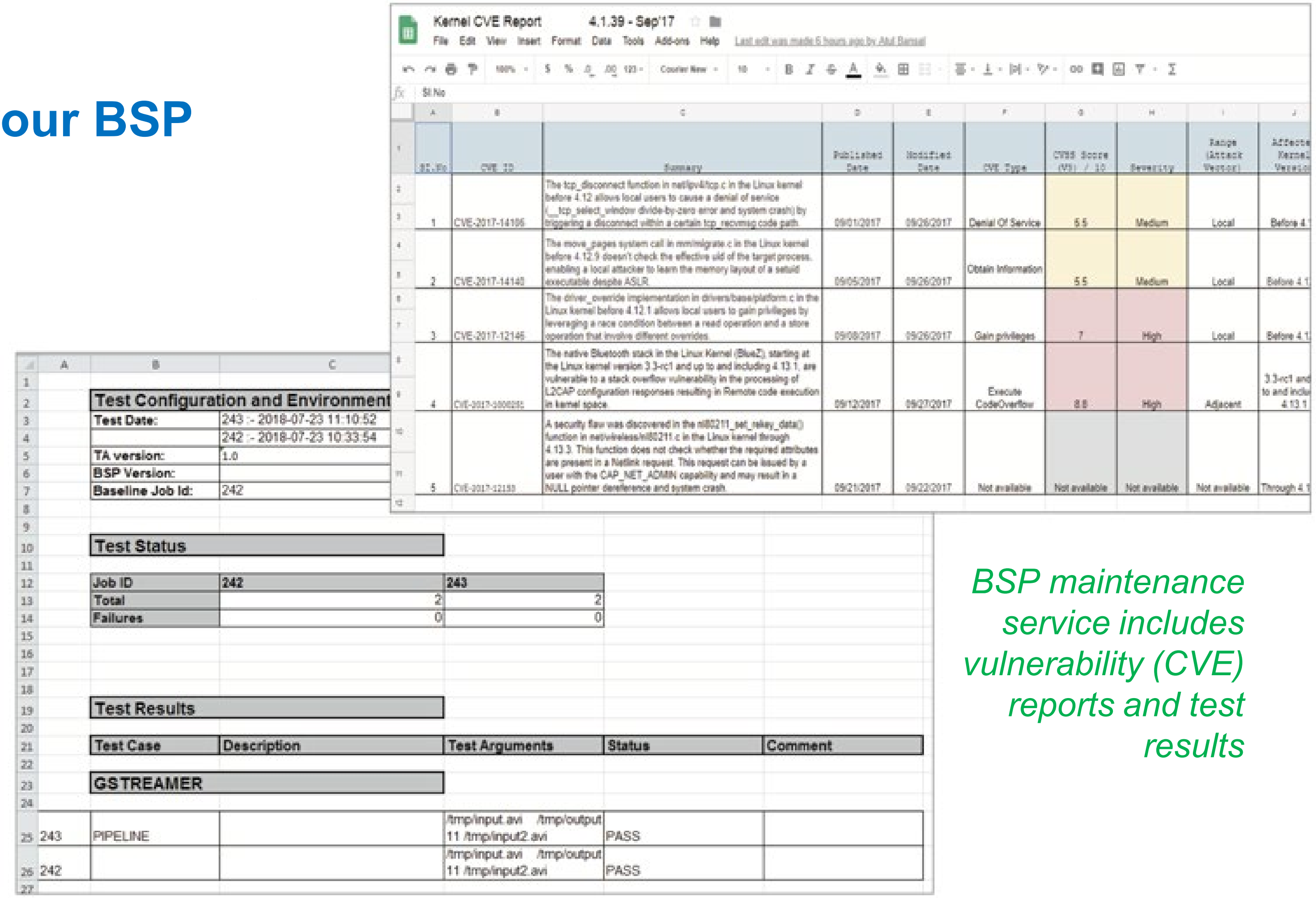
- Generic driver tests
- Support for manual and automated tests

Board Farm Cloud

- Automated deploy
- Automated test runs
- Reports

Introducing: BSP Maintenance Service

- Turnkey service that maintains your BSP throughout its lifecycle
  - Keep pace with updates
  - Maintain product security
  - Cut BSP maintenance costs
- Focus your resources on development & differentiation
- Provides visibility and control at all times



Kernel CVE Report 4.1.39 - Sep'17

fx Sl.No

Sl.No	CVE ID	Summary	Published Date	Modified Date	CVE Type	CVSS Score (V3) / 10	Severity	Range (Attack Vector)	Affecte Kernel Versio
1	CVE-2017-14106	The tcp_disconnect function in net/ipv4/tcp.c in the Linux kernel before 4.12 allows local users to cause a denial of service (__tcp_select_window divide-by-zero error and system crash) by triggering a disconnect within a certain tcp_recvmsg code path.	09/01/2017	09/26/2017	Denial Of Service	5.5	Medium	Local	Before 4.1
2	CVE-2017-14140	The move_pages system call in mm/migrate.c in the Linux kernel before 4.12.9 doesn't check the effective uid of the target process, enabling a local attacker to learn the memory layout of a setuid executable despite ASLR.	09/05/2017	09/26/2017	Obtain Information	5.5	Medium	Local	Before 4.1
3	CVE-2017-12146	The driver_override implementation in drivers/base/platform.c in the Linux kernel before 4.12.1 allows local users to gain privileges by leveraging a race condition between a read operation and a store operation that involve different overrides.	09/08/2017	09/26/2017	Gain privileges	7	High	Local	Before 4.1
4	CVE-2017-1000251	The native Bluetooth stack in the Linux Kernel (BlueZ), starting at the Linux kernel version 3.3-rc1 and up to and including 4.13.1, are vulnerable to a stack overflow vulnerability in the processing of L2CAP configuration responses resulting in Remote code execution in kernel space.	09/12/2017	09/27/2017	Execute CodeOverflow	8.8	High	Adjacent	3.3-rc1 and to and inclu 4.13.1
5	CVE-2017-12153	A security flaw was discovered in the nl80211_set_rekey_data() function in net/wireless/nl80211.c in the Linux kernel through 4.13.3. This function does not check whether the required attributes are present in a Netlink request. This request can be issued by a user with the CAP_NET_ADMIN capability and may result in a NULL pointer dereference and system crash.	09/21/2017	09/22/2017	Not available	Not available	Not available	Not available	Through 4.1

Test Configuration and Environment

Test Date:	243 :- 2018-07-23 11:10:52
	242 :- 2018-07-23 10:33:54
TA version:	1.0
BSP Version:	
Baseline Job Id:	242

Test Status

Job ID	242	243
Total	2	2
Failures	0	0

Test Results

	Test Case	Description	Test Arguments	Status	Comment
	GSTREAMER				
243	PIPELINE		/tmp/input.avi /tmp/output 11 /tmp/input2.avi	PASS	
242			/tmp/input.avi /tmp/output 11 /tmp/input2.avi	PASS	

BSP maintenance service includes vulnerability (CVE) reports and test results

What Is Included in the Service Package

- A subscription to Vigiles Prime
  - Security & vulnerability notification and reporting tool for monitoring your software
- Complete BSP update (software release) twice a year (by default / cadence can be changed)
  - Minor kernel version upgrade for security and bug fixes
  - User space security patching & package updates
  - Two releases per year on a mutually agreed timeline
  - Only mutually agreed upon items will be integrated
- Each update is validated and tested on the customer’s hardware
  - Release notes and test reports included with each update
  - Customer provided HW is maintained in our board farm
- BSP is maintained on a secure, private, bidirectional Git server
  - upload/download sources and changes
- In the event something critical happens between updates…
  - On-demand update for emergency security fixes (one per year included)

The Hidden Costs of BSP Maintenance

Tasks	1st Board	3 Boards*	5 Boards*
Monitoring	$20k	$25k	$30k
Finding & Applying Patches Finding Fixed Versions & Upgrading Versions	$38k	$50k	$60k
Testing 2 Releases Per Year	$32k	$75k	$120k
Total	**$90k**	**$150k**	**$215k**

BSP Maintenance

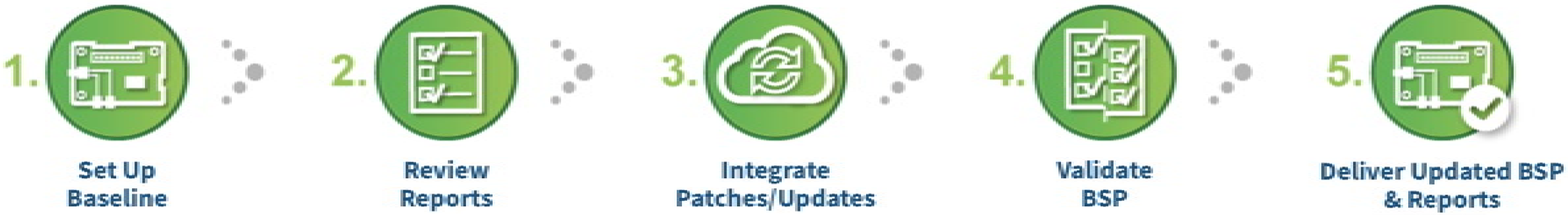
Do It Yourself: **$150,000 / year**
Timesys: **$75,000 for 3 boards**

**Assume more than 75% overlap in Software components and kernel configurations*

How to Engage Pro-Support to Maintain Your BSP

- Customers sign up
- Hardware and BSP are provided to NXP
  - *NXP will use this to establish a baseline test report*
- Pro-Support will periodically review the recommended updates to include in the upcoming release
- The updated BSP will be tested on the customer’s platform and delivered twice a year
  - *Including release notes and test report*

Special Introductory Price

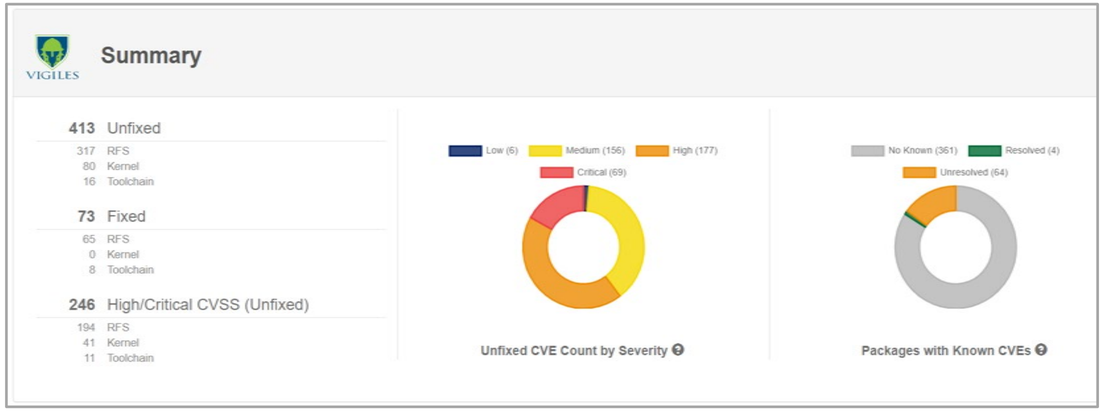


1. Set Up Baseline
2. Review Reports
3. Integrate Patches/Updates
4. Validate BSP
5. Deliver Updated BSP & Reports

BSP Maintenance Solution: Stretch Right

Turnkey service that maintains your BSPs throughout the product life cycle

- Extends security beyond development into production deployment
- Cuts BSP maintenance costs by 50% +
- Applies latest updates for improved stability and security
- Simplifies vulnerability tracking and fixing with auto notification and suggested fixes
- Performs updates and tests for your hardware
- Gives full visibility and control at all times
- Integrates with your dev process with shared private Git and full release notes
- Supplies updates you pick on your schedule
- Permits you to focus dev cycles on new products & enhancements



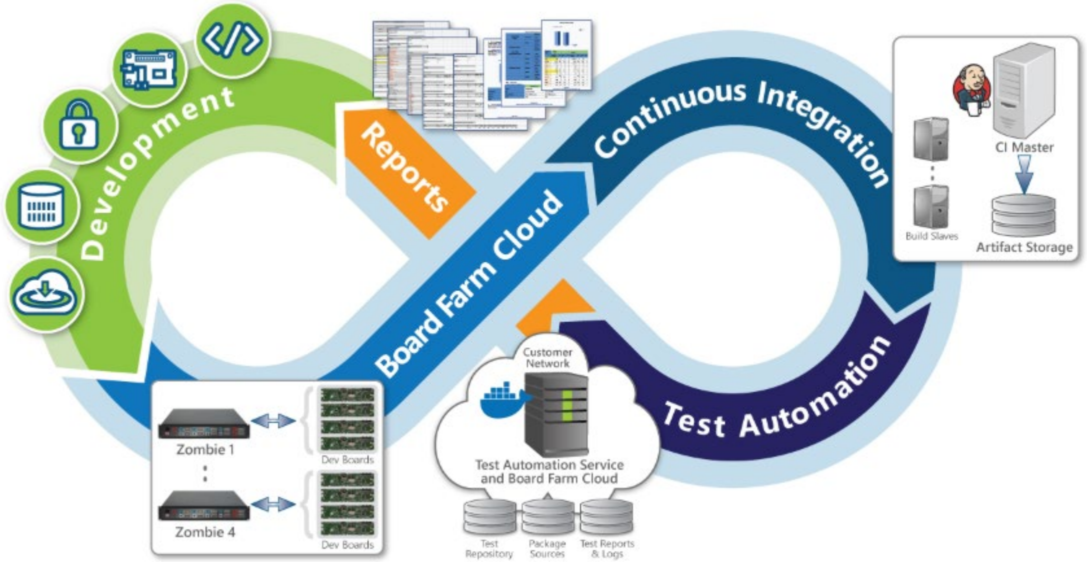
VIGILES

Summary

413	Unfixed
317	RFS
80	Kernel
16	Toolchain
73	Fixed
65	RFS
0	Kernel
8	Toolchain
246	High/Critical CVSS (Unfixed)
194	RFS
41	Kernel
11	Toolchain

Unfixed CVE Count by Severity

Packages with Known CVEs



For More Information and to Become More Secure

Contact us at Vigiles@nxp.com

Or

Use this link to go to the BSP Lifecycle Maintenance page on NXP.com

Thank You!

NXP

SECURE CONNECTIONS
FOR A SMARTER WORLD