GUIDE

MARCH 2016

Application Security Buyer's Guide

A practical handbook for selecting the right tools and vendors

TechBeacon.COM

Table of contents

Application Security Buyer's Guide 03

Securing code made harder by changing development 4
Selling the secure development lifecycle 5
Take stock and just get started 6
Three classes of tools for your developers 7
SAST, DAST, IAST: A lot of choices 14

Model Request for Proposal (RFP) 15

Statements about your current status 16
Questions for vendors 17

Application Security Buyer's Guide

More companies are seeing the benefits of secure development and relying on application security tools to lock down their application portfolio. This guide takes an in-depth look at the tools and the best way to create a mature and secure development program.

In 2015, Adobe patched more than 300 security vulnerabilities in its Flash player, a ubiquitous piece of software that enables interactivity and multimedia in browsers on the Internet. This was not just an act of good software hygiene— Flash vulnerabilities have real-world consequences. They have become the most popular way to exploit and take over users' machines, accounting for eight of the ten most used exploits to compromise systems, according to data science firm Recorded Future.

The travails of Adobe show the danger of software bugs and the importance of application security testing. Yet exorcising bugs is typically given short shrift within companies. It is a hard task, even when a company has focused on eliminating software flaws in its products.

This guide will briefly describe why application security presents special challenges for organizations, even those that have made significant investments in network and endpoint security. It will describe these challenges in terms of actual experiences teams have faced, and it will include steps organizations can take to move toward app sec best practices. It will also describe the essential classes of technologies that have proved useful in support of those practices.

And it concludes with a separate step-by-step guide to creating an RFP for readers actively seeking app sec vendors and tools.

Vendors have moved to make application-security testing easier; they're automating much of the manual work and adding security expertise... Yet a number of problems continue to plague the sector.

Securing code made harder by changing development

While most businesses have invested in network and endpoint security, taking measures to ensure that applications are both designed and coded in a secure way is much less common. Those measures require both management and developer approval. In a recent survey, 56% of corporate security teams had a security information and event management (SIEM) system in place, while only a third used either static or dynamic application security testing, according to data from analyst firm 451 Group.

"We are overspending in antivirus and firewall and not enough in applications, which is what all the attacks are targeting," says Jeremiah Grossman, co-founder and former CEO of Whitehat Security, a web-application security firm.

That's beginning to change. Companies are increasingly signing on to secure development. Over the past decade, vendors have moved to make application-security testing easier; they're automating much of the manual work and adding security expertise to the software to reduce the reliance on security professionals.

Yet a number of problems continue to plague the sector:

- False positives continue to create work for security professionals and developers who have to rule out the alert
- Many types of vulnerabilities still escape detection
- More accurate systems tend to take longer to run, slowing development

Because of the complexity of tools, the difficulty in getting developers to add steps to their coding process, and an overall lack of focus on security, most companies are only dipping their toes into the water.

The situation is made more complex because developers are changing the way they are creating software. Traditional waterfall development practices engineered the application up front, with design mandates that flowed down to the coders. Developers had months, even years, to produce working software, which gave the security team natural breaks during which they could review the code. But newer agile and DevOps models using short- or continuous-development cycles aim to create a fast feedback loop on the order of days, if not less. The shortening development cycle is a problem for security teams looking to implement application-security testing.

Selling the secure development lifecycle

In 2015, a transoceanic shipping firm regularly faced pirates boarding its ships traveling near Singapore toward the Indian Ocean. The pirates knew which containers to open and where to find the most valuable items, typically gems and jewelry. How did they know where to look? Verizon, which responds to cyber-security incidents, found that the company "used a homegrown CMS [content management system] to manage inventories," and that the hackers had used a vulnerability in the system to get access to bills of lading.

60% JAVA **30%** .NET & RUBY

Applications being developed at Aetna

While companies typically do not have to deal with sea pirates, the incident underscores that vulnerabilities can cost real money. For those companies that make the effort, the savings can be significant.

Healthcare company Aetna, for example, has more than 3,000 applications in development, two-thirds of which are written in Java, with much of the remainder written in .NET and Ruby. Keeping the developers of those applications focused on security is tough, but it can result in significant cost savings, says Jim Routh, the company's chief information security officer.

"Our whole software security program is predicated on the objective of reducing the total cost of ownership of the application portfolio," Routh says. "We try to prevent as many defects as we can, and we try to fix as many bugs as well as we can, and that ends up creating a productivity gain of 20 to 50 percent."

Take stock and just get started

To determine which path is right, companies first need to take stock of their current software security practices. Unfortunately, many companies don't know where they stand, says Chris Wysopal, chief technology officer and co-founder of application security firm Veracode.

"Sometimes you talk to the security people and they say their developers are doing application security," Wysopal says. "The security team thinks something is happening, but in reality, the tools and processes are not being used by the developers."

Too often, companies will buy an application security testing tool or subscribe to a service, vet two or three applications, and then stop. Problems arise when they try to use the results produced by the tools or try to expand their coverage to their entire portfolio, says Dan Cornell, chief technology officer for the Denim Group, a security-testing consultancy.

"Even after you get an application under testing, often no one knows what to do with the results," he says. "The interpretation, the prioritization—all of that is a real challenge."

Maturity takes time

Improving application security is an ongoing, massive undertaking, one that companies should not think of as a time-boxed project. Developing a mature security development lifecycle requires time and many iterations. In particular, getting teams working together and using the available tools is, in some ways, more critical than selecting the right tools. Strong programs are built on accountability and communication, says Mike Spanbauer, vice president of research at NSS Labs.

"Enterprises must find a way to work closely between those two teams," he says. "Often the dev teams says, 'That's not my problem,' that it's the security team who's responsible to fix that. But they can't protect the environment optimally unless they know which hooks, libraries, and code are used by the developers."

"[Security] can't protect the environment optimally unless they know which hooks, libraries, and code are used by the developers."

Mike Spanbauer, vice president of research, NSS Labs

For many companies, a good first step is to take stock of their application portfolio and determine which third-party libraries and components are used by developers. Development teams may not adequately track the versions of the libraries embedded in their code, but they should know when a vulnerability is released in a library that they use in their development, and which applications are affected.

"Being able to quickly index which library's components you have allows you to then leverage other information sources, [which means] you don't have to do the homework yourself," says Spanbauer. "You can quickly determine whether that piece of vulnerable software is one that you are using."

Expanding coverage

Once companies get testing tools running for a few applications, they should broaden their efforts to the company's full portfolio. As part of the expansion, companies should track a few important metrics, such as what portion of their application portfolio is under testing, as well as what percentage of vulnerabilities are either remediated or have a workaround, according to Denim Group's Cornell.

"It is less about the specific technology and more about just getting that workflow operating," Cornell says.

As part of this step, companies need to take stock of all their applications and who is lead developer on each. While top-down requirements for development groups to use specific tools can work, often it works better to offer the developers a specific menu of security tools and options and let them use whatever tool best fits their purpose.

Three classes of tools for your developers

Any company developing software as a product, a service, or for internal use should be focusing on software security, but there is no one perfect way to embark on developing, or improving, a secure development lifecycle.

BACK TO
THE INDEX

Some companies need to focus on their developers, adding static application security testing (SAST) to the development cycle to catch potential flaws early. Other organizations may want to prove their need by using dynamic application security testing (DAST) or penetration testing to show how vulnerable their applications are. Interactive application security testing (IAST) uses agents and additional software libraries to collect data from running applications that can then reveal vulnerabilities. Companies that want to "virtually patch" their applications can lock down their portfolios using some form of application firewall or a newer technology, such as runtime application self-protection (RASP), a form of IAST.

The following information outlines the benefits and potential drawbacks of each of the three technology classes noted above. (A list of vendors and their products is included; and a model request for proposal for contacting vendors is provided later in this guide.)

1. Static Application Security Testing: Eliminate vulnerabilities early

The front line of secure code development tends to be static application security testing (SAST) tools—software programs that scan source code to find known patterns of vulnerabilities. These tools are increasingly being provided to developers as the first step toward weeding out the most obvious vulnerabilities from their code.

SAST tools have a bad reputation for producing too many alerts for minor software flaws and unexploitable defects and for finding only narrow classes of bugs. Nonetheless, the tools are an important part of any secure development lifecycle because they can be integrated into the development environment, preventing developers from making fundamental security errors that could produce vulnerabilities and teaching them to avoid making similar mistakes in the future.

Eliminating vulnerabilities during development saves money. Estimates range from a low of 20% up to a high of 100% savings, if a defect is caught at the initial design level. "The cheapest place to find bugs is early in the secure development lifecycle," says Daniel Kennedy, research director for information security at the 451 Group.

"The cheapest place to find bugs is early in the secure development lifecycle,"

Daniel Kennedy, research director for security, 451 Group

BACK TO THE INDEX

Integrate automated code checking into development cycle

A very important quality of any SAST tool is how well it integrates into the company's existing development environments. Most tools support the major web languages, Java and .Net. Static tools generally also support some form of C, C++, or C#.

In addition to supporting the language that your development team uses, the tools have to seamlessly plug into the most common integrated development environments (IDEs).

"The developer is extremely busy," says IBM VP Ravi Srinivasan, who's responsible for network security offerings. "They don't want to have to go to a separate program to run application security testing."

SAST is not just about catching bugs

A SAST system's most common user should be the developer. The best tools provide feedback to teach the developers how to use secure software patterns, avoid code libraries deemed to be too insecure or infrequently patched, and explicitly explain how to repair any potential vulnerabilities.

Providing developers with simple-to-use static analysis tools that can check their code quickly and provide easy-to-understand feedback pays dividends down the road. Aetna, for example, found that developers using static analysis tools learned to produce more secure code.

"What we learned is that the defect density drops significantly when developers use static analysis tools," Aetna's Routh says. "They are actually learning how to avoid defects on their next project, and it is the most valuable tool in teaching developers secure coding practices."

"The defect density drops significantly when developers use static analysis tools."

Jim Routh, chief information security officer, Aetna

BACK TO THE INDEX

Strike a balance between SAST false positives and developer goodwill

Today's application security testing tools are getting more comprehensive, but they still require developers and security professionals to weed out false positives. In some way, the greater depth of testing presents its own problems, says Denim Group's Dan Cornell.

"A lot of static vendors were trying to get the most results, but what people realize now is that there is a tax that comes along with all the stuff you found," he says. "So finding more stuff is not necessarily good. If I find too much stuff, I have to wade around in all the results to find what is really important."

The issue can be critical for companies focused on an agile or DevOps model, who may find that static analysis tools take too long to run to be used in daily development, Aetna's Routh says.

"The longer it takes to run, the more difficult it is and the longer it takes security people to weed out the false positives and the less integrate-able it is with the development cycle," he says.

2. Dynamic Application Security Testing: Finding exploitable vulnerabilities

Companies can take a more strategic approach to testing by using DAST. Also known as "black-box testing," dynamic analysis tests for various types of vulnerabilities against running applications.

The choice between adopting static or dynamic analysis tools first depends on a company's situation. Static analysis tools give developers feedback and educate them at the same time. Dynamic analysis tools can give security teams a quick win by immediately finding exploitable vulnerabilities.

In most cases, companies should run both. Static- and dynamic-analysis tools plug into the development process in different places. Static tools should be run as often as is practical and give feedback directly to the developer, allowing managers and the security team to monitor the progress of developers in eliminating bugs. DAST should be done less often and by dedicated security and quality-assurance professionals.

Dynamic analysis tools can give security teams a quick win by immediately finding exploitable vulnerabilities.

BACK TO THE INDEX

While DAST is typically suited to a waterfall development approach, agile and DevOps development can benefit as well, says Aetna's Routh.

"Dynamic scanning is something that you apply after you have a runtime version of the software," he says. "Dynamic scanning is typically done by the QA teams. In a Scrum team, there may be a designated QA team that is part of each sprint, but they are not the developers."

Dynamic analysis can help find assets

The first step for most companies is to make sure they know which applications are running on their network. Companies that embark on a new application-security initiative can tie network scanning into the DAST process. This requires first searching the network for applications, then testing those applications for vulnerabilities. The process can help security teams stay on top of unknown and rogue applications, says Frank Catucci, application-security manager for Qualys, a network and application security firm.

"A lot of customers who come to us say they know what they need for applications security, but they have so much that is out there running, there may be Web services that they are not be aware of," he says.

By asking a few simple questions, companies can better understand their software security status quo. Security groups should first figure out how many web applications they have running, says Jeremiah Grossman.

"Most companies do not know what they own, and that is a major issue," he says. "Down the road, when they get hacked, when they get compromised, it is almost always because of a vulnerability in a web site they did not know about."

"Most companies do not know what they own... When they get hacked, it is almost always because of a vulnerability in a web site they did not know about."

Jeremiah Grossman, co-founder and former CEO, Whitehat Security

Build DAST into the quality assurance phase

While static code analysis can find bad patterns in code and teach developers more secure coding techniques, DAST is about catching exploitable vulnerabilities before they get into a production environment. Development groups and security teams should build a dynamic testing phase into the pre-release quality assurance phase of any secure development lifecycle.

BACK TO THE INDEX

The goal is not about finding every vulnerability, but to find the most exploitable issues, says Jeremiah Grossman.

IAST promises to catch attacks that the other approaches cannot.

"You don't want a mess of vulnerabilities in production, so you use SAST to rid yourself of those as much as you can," he says. "But not all vulnerabilities are available to attackers; there are a lot of vulnerabilities that do not pose a threat. That is what DAST is for—finding vulnerabilities relative to the bad guys, where there are no more rules."

Shorten testing and remediation time

Testing takes time, and making sense of the results of that testing can take even more time. While false positives are less of an issue for dynamic application security testing, security teams and developers should focus on evaluating each product's ability to quickly test for a required set of potential vulnerabilities and produce reports that succinctly tell developers how to fix the problems.

Time becomes an even bigger issue for development groups focused on a DevOps model of software development and deployment. Such groups have to automate more and push the testing to earlier in the process, making time a critical factor, says Adrian Lane, chief technology officer for Securosis, a security consultancy.

"It changes the way we test, and it forces companies to build a lot of their own infrastructure," he says. "So a big question is whether you can use an API to script and automate the testing, not just integrate with the developer systems."

3. IAST, including RASP: Protecting code against exploits and known bugs

While static code analysis arrived about two decades ago and dynamic analysis has become popular over the last decade, a new approach—known as interactive application security testing (IAST) or "glass-box" testing—promises to catch attacks that the other approaches cannot. By running an agent, IAST allows companies to collect event data from their running applications for analysis.

BACK TO THE INDEX

Jeff Williams, chief technology officer of Contrast Security and a pioneer of the approach, argues that both static and dynamic analysis fail to catch many types of vulnerabilities that affect applications. IAST can help to better secure the software, he says.

"Bottom line for me is that dynamic and static tools have been around for over a decade without improving and are not likely to improve," he says. "It's a huge problem that prevents doing application security at scale and leaves us all vulnerable."

The promise of software agents

The key of interactive analysis is turning the application into a tool that also works for security. Either by installing software agents on an application server, or by instrumenting the application at development time, interactive analysis techniques allow the collection of data on application and security events.

The data can detect software flaws that other techniques would miss. In addition, combining agent-based instrumentation of applications with code analysis and dynamic scanning can help reduce false positives and give developers a better idea of where vulnerabilities exist in their code.

"I've always disliked the idea that security makes things less efficient," Contrast's Williams says. "Security is an enabler. Done right, application security technology speeds software development and operations by providing continuous, accurate, useful information in real time and at scale."

"Security is an enabler. Done right, [it] speeds software development and operations by providing continuous, accurate, useful information in real time and at scale."

Jeff Williams, CTO, Contrast Security

Got legacy apps? Protect them with RASP

The same agents and libraries that can be used to detect the signs of vulnerabilities in programs can also block bad behavior—a likely sign of an attack. The use of runtime application self-protection (RASP) technology can prevent attacks and deliver immediate benefits to companies. For security teams that need to protect applications that may not be actively developed, an application firewall or runtime application self-protection (RASP) agent may work well. RASP is best thought of as a specific type of IAST.

BACK TO THE INDEX

"RASP sits inside applications and can monitor and block vulnerabilities," says Maria Bledsoe, a senior manager with HP Enterprise's security products group. "RASP is a compensating control."

Pilot an IAST program

Because IAST is a relatively new technology, companies should pilot a tool to determine if it's right for their development programs. More mature security teams are already piloting IAST efforts. Aetna, for example, will likely deploy the technology in a year.

"We see it as something that we are definitely doing," says Jim Routh, CISO of Aetna. "So we are going through a process of evaluating the products. The tools are relatively immature, but they are promising."

SAST, DAST, IAST: A lot of choices

Most security experts agree that there is a role for all three technologies: static, dynamic, and interactive analysis.

While static analysis aims to help developers produce better and more secure code, dynamic analysis heads off exploitable vulnerabilities before they are released. IAST instruments an application so that information about potential malicious activity can be gathered while the software is running. And some IAST systems also block the attack, which is the function of runtime application self-protection, or RASP, technique described above.

In the end, how a company proceeds in adopting application security testing tools depends on how mature its efforts are in creating a secure development lifecycle, what type of software is under development, and the resources the company can dedicate to the effort.

In implementing security development lifecycles at six different organizations, NCR's enterprise security architect Nir Valtman has learned that progress is what's most important.

"Whether you have SAST or DAST or IAST, it doesn't matter how many bugs you have at the moment," Valtman says. "The only thing that matters is the trending chart—reducing the number of bugs that you have."

"SAST or DAST or IAST … The only thing that matters is the trending chart—reducing the number of bugs that you have."

Nir Valtman, enterprise security architect, NCR

BACK TO THE INDEX

Guide to creating an RFP for application security testing tools and services

Companies looking to query vendors about potential tools and services for application security can use this guide in creating a request for proposal.

While there are a lot of questions in this model RFP, companies should choose no more than ten that are critical to their business. Rather than waiting for the perfect product, companies should find the one that best matches their most important criteria.

While there are a lot of questions in this model RFP, companies should choose no more than ten that are critical to their business.

Why narrow your RFP to a ten or fewer questions? Because organizations that send out a laundry list of requirements run the risk of distorting their acquisition process. There are dozens and dozens of potential requirements, and in their proposals vendors will likely describe how their product meets each of them. You will have too much to sort through, and comparing one proposal to another will become an arduous if not impossible task. Companies should instead focus on the 5 to 10 requirements that are most important for their implementation and compare vendors on the basis of those issues.

Other good RFP examples you might explore include:

- OWASP's Application Security Verification (RFP-Criteria)
- Veracode's Sample RFP for Application Security Scanning Tool Selection

BACK TO THE INDEX

Statements about your current status

Typically, an RFP includes a few up-front statements to clarify the security context for vendors. Consider writing a statement of goals and concerns and a statement of resources, as described below.

Often, the company's security effort is not mature enough to state firm goals. Instead, companies should focus on highlighting their security problems.

Statement of goals and concerns

Many model RFPs start with a statement of a company's goals—in this case, security goals. But often, when companies embark on an initiative to improve the security of their development process, the company's security effort is not mature enough to state firm goals. Instead, companies should focus on highlighting their security problems.

So, state your goals for application security testing if you can do this, but most important is that you clearly state the greatest concerns for company management.

Goals should include:
Services

- Manual code review
- Manual penetration testing
- Analysis of security design and architecture
- Threat modeling

Products or services

- Automated code review (static analysis)
- Automated penetration tests and vulnerability assessments (dynamic analysis)
- Automated analysis of instrumented applications (interactive analysis)

Because vendors and service providers will often act in the role of a partner or provider, companies should also state their problems, which will aid vendors in determining if unasked-for services or products may be a better fit. Here are examples of some potential problems:

BACK TO THE INDEX

- Our SIEM system detected significant vulnerabilities in our applications that we need to remediate
- A penetration test has found security problems in an Internet-facing application
- We only partially cover our application portfolio
- Our developers continue to make the same coding mistakes that lead to vulnerabilities

Companies should also state their problems, which will aid vendors in determining if unasked-for services or products may be a better fit.

Statement of resources

In addition, companies have to determine what resources and capabilities they have to bring to bear on the problem:

- How much security expertise do they have and how many full-time employees (FTEs) can they dedicate to application security?
- How many websites or applications do they have to secure?
- How many developers will need to work with the tools?

Questions for vendors

The remainder of your RFP should focus on the dozen, or fewer, questions that get to the heart of your requirements. Use one or more queries from each of the following six categories to build out your RFP. Obviously, you should tailor your queries to fit your situation.

1. Vendor information

Purpose: To gather basic information about the vendor's company.

- Describe your company, its history, and its security focus.
- Who is the primary contact for sales? Who is the primary contact for technical questions?
- Do you have reference clients that can be contacted?

BACK TO THE INDEX

2. Products, services, and expertise

Purpose: Discussion of products and services offered; the vendor's approach to application security testing; and what innovations the vendor offers.

- ☐ List your products and their coverage of languages and development environments. What is their primary category: SAST, DAST, IAST, or RASP?
- ☐ Is your product or service offered on-premises, as a cloud service, or as a hybrid solution?
- ☐ What are the strengths of your product or service? What is its valid flaw (true positive) rate?
- ☐ What are the weaknesses of your product or service? What is the false positive rate?
- ☐ Which applications and programming languages can be tested? What development environments do you support?
- ☐ What are the core features or innovations that separate your product from others?
- ☐ How often do you release new versions or update the software for new classes of vulnerabilities?

Ask your vendor: What are the strengths of your product or service? What is its valid flaw (true positive) rate?

3. Deployment

Purpose: These questions are designed to gauge the level of investments—in money and manpower—needed to deploy and maintain a system. Flexible deployment models are in demand.

- ☐ Does the application security testing require an on-premises capital investment?
- ☐ How quickly does it take to deploy the system? How long does it take to add an application to the security tests?
- ☐ Can geographically distributed development groups easily use the product or services? Please describe how.
- ☐ Does deployment require a consulting engagement? What other services do you offer before, during, or following deployment?
- ☐ What is the pricing model used? Please provide estimates for the following use case. (DESCRIBE A TYPICAL USE CASE.)
- ☐ Does deployment require a support contract? What are the terms of your support contracts?

BACK TO THE INDEX

4. Operations

Purpose: This section characterizes the day-to-day operations, costs, and resource requirements of running the software or service.

- How long does the tool take to run? Will it slow down my development process? Is it appropriate for DevOps or agile development?
- Can the tool be run by individual developers? Is it run as part of the development process or as part of the QA process?
- How easily can the product be managed? Approximately how many full-time employees (FTEs) does it take to manage?
- How can learning (data output and analysis) from the system be incorporated into our development lifecycle?

How easily can your product be managed? Approximately how many fulltime employees (FTEs) does it take?

5. Reporting, interoperability, and integration

Purpose: The major deliverable from application security testing systems is vulnerability information. This section focuses on how that information is delivered.

- What types of reports do you offer and who are the consumers (developers, security, management) of each one?
- What information can be included in each report? Do vulnerability reports give detailed instructions on how to mitigate the vulnerability in a way that can be understood by a developer?
- Do the reports provide historical trends and key benchmarks regarding the remediation of vulnerabilities?
- Does it provide such data in a format easily digestible by security information and event management (SIEM) systems and other security tools?
- Do vulnerability reports describe the specific steps needed for developers to remediate discovered vulnerabilities and steps to detect if the vulnerability has been closed?
- Does your product use or support APIs to directly communicate results with other security and IT management technology?

BACK TO THE INDEX

6. Privacy and data security

Purpose: Source code is a sensitive business asset. This section asks vendors how that information is protected.

- How is source code handled and secured?
- What types of encryption do you use and what data is encrypted?
- Do you offer a private key system where the encryption keys can be managed by our company?
- Do you keep data on our usage of the tools and other corporate sensitive data? What sort of security and privacy guarantees can you make?

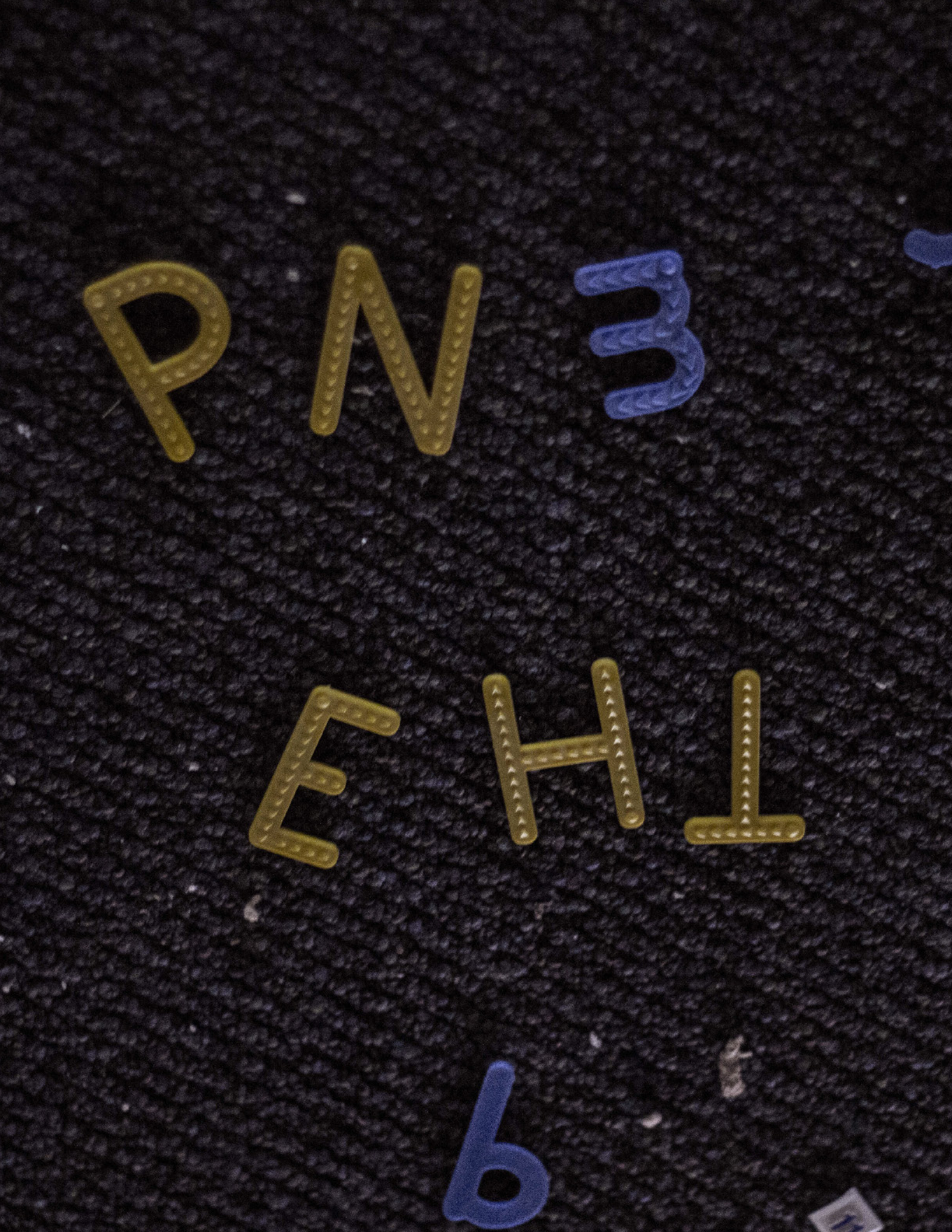
Pare your questions down to 10 critical requirements... and the proposals you receive will be much easier to compare as you narrow your candidates.

Focus on your most critical needs

As you think through the information presented above to create an RFP, be sure to tailor it to your specific needs. Don't just send out all the questions provided in the six areas of enquiry! As noted earlier, pare your questions down to 10 critical requirements for your desired application security capabilities. Vendors will appreciate your laser focus, and the proposals you receive will be much easier to compare as you narrow your candidates.

Good luck!

BACK TO THE INDEX


PN3
EHL
b

TechBeacon