



Security challenges

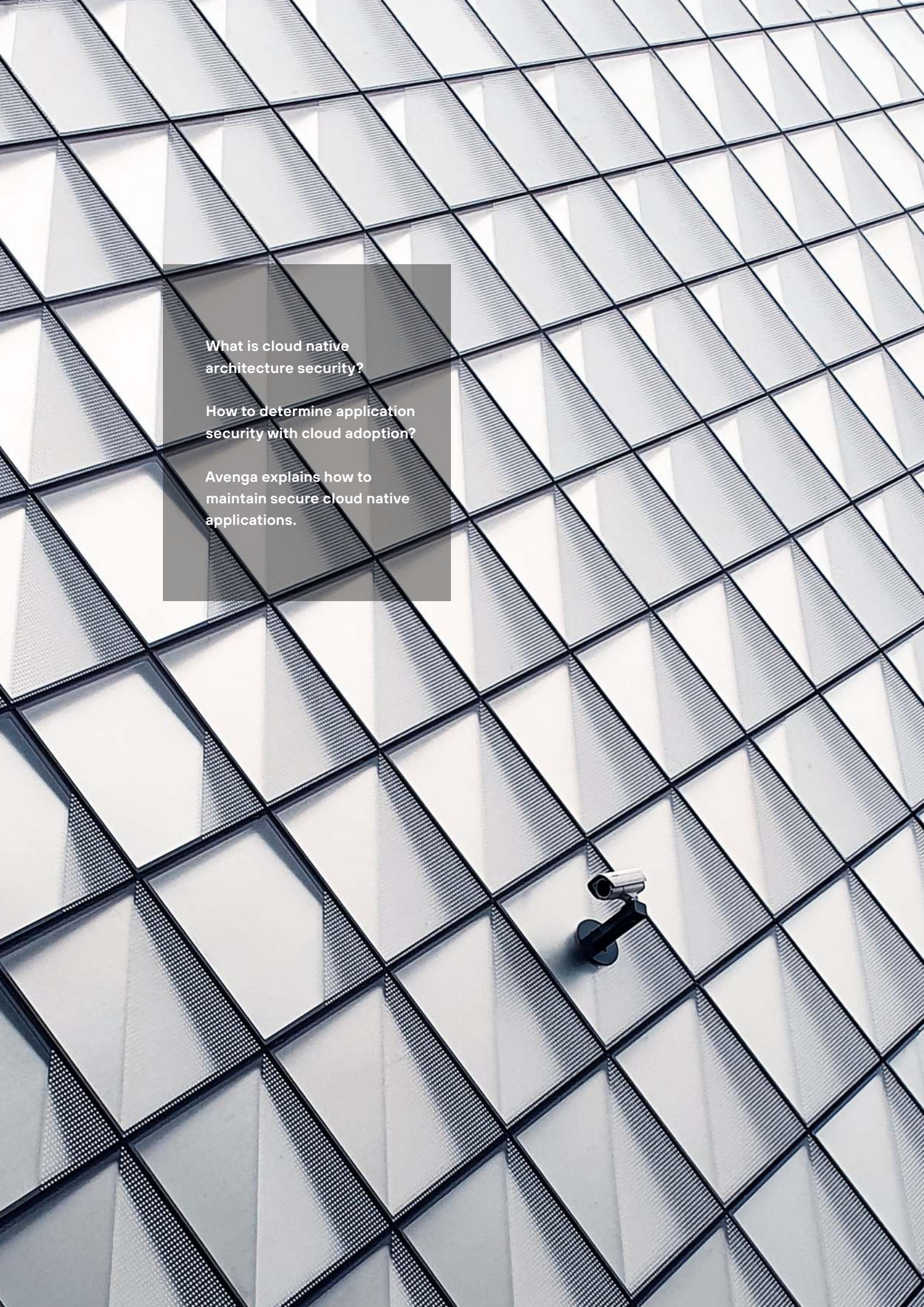
in cloud native architectures

Table of contents

1. More flexibility calls for more responsibility	04
2. Kubernetes and security	05
3. Containers and security	06
4. Networking	07
5. Monitoring	08
6. Automation to the rescue?	08
7. Security culture	09
8. If you think you're done...	09
9. The future of secure cloud native solutions	10
10. Why Avenga?	11

Read the original article here:

<https://www.avenga.com/magazine/security-challenges-cloud-architecture/>



What is cloud native
architecture security?

How to determine application
security with cloud adoption?

Avenqa explains how to
maintain secure cloud native
applications.

1. More flexibility calls for more responsibility

New options to build digital solutions faster are great and without them the subsequent waves of digital transformation can be much slower.

One of these options is a cloud native architecture, which is often associated with hybrid cloud architectures that enable services to run both on-premise and in the public cloud together. The cloud native is also considered a way to avoid cloud vendor lock, as containers and orchestrators are creating abstraction which makes it easier to migrate solutions between different clouds and on-prem infrastructures.

DevOps, Kubernetes, and Docker are known by almost everyone in the IT industry now. The majority of IT organizations continue optimizing their cloud native infrastructures with new technologies, tools and management methodologies.

But what about security? What is new in this cloud native world? Do old golden security rules still apply?



2. Kubernetes and security

One of the things, which you can not forget, is that Kubernetes is running on top of the **operating system (OS)**. The OS itself is vulnerable to attacks and its security needs proper constant maintenance. There's no escape from any layer of the abstraction. Any lack of attention always provides a great opportunity for hackers.

Within the **Kubernetes orchestrator**, all its components are software components with their own life-cycle and set of vulnerabilities. Upgrades of Kubernetes engines are not what come to mind in the first place when building cloud native foundations, but they are inevitable.

Upgrading entire clusters without any downtime is possible, albeit a challenging effort. Sometimes companies choose the option to do it outside of production windows (if possible). Upgrading orchestrators, kubelets and other components may cause application instability, so we will always trade some stability for security in this case.

First of all, the components need to be **upgraded** to avoid stale, deprecated, and unsupported elements, and then, all those components have to be updated regularly to address known vulnerabilities. The Kubernetes team is publishing a list on the internet group of which security vulnerabilities are addressed by each patch and version.

In the case of Kubernetes running on the public cloud, this part of the job is the responsibility of the cloud provider.

Security management of a Kubernetes management API is another challenge to be addressed. **Role Based Access Control** should be implemented, thus allowing different users to perform different actions and access tight resources. Almost nobody needs full administrative access to the orchestrator in order to perform the majority of operations.

Namespaces are the great Kubernetes mechanisms to isolate sensitive workloads and resources.

Vulnerability scanning for running pods is also recommended to prevent any vulnerable pods from running without patching, after a new security problem is found.

Each of the many kubelets running on multiple nodes should only allow authenticated traffic, anonymous access should be denied.

And last but not least, the default configuration of Kubernetes is insecure and always requires hardening for production, in fact, for any environment.

3. Containers and security

Containers mean a lot of flexibility, with much less overhead compared to traditional virtual machines. But, one of the costs of this optimized efficiency is lowered security. Containers are much less secure than virtual machines, as the separation between them, done on the Linux kernel level, is not as tight as the one done on the virtualization level. So, it's more likely that one container may escape the boundaries than in the case of VMs.

Containers, well, they contain packages of different software libraries and all of those libraries may be vulnerable. **Scanning containers** as part of continuous integration and delivery pipelines is highly recommended.

Containers should come from **trusted and reliable sources**, so as to avoid using insecure containers as the base for the applications. Strange ad hoc container image registries should be avoided.

The less there is in the container the better, because it's one of the principles of the **distroless containers'** popularity. It's worth considering as an option to both improve performance and reduce storage requirements, while at the same time greatly reducing the attack surface.

Done? What else? Many of the components are scripts or compiled libraries from custom written code; and without surprises, as the code may contain security issues. There's no way around good architecture, designing proper software, and avoiding common security antipatterns that are defined in standards such as the Open Web Application Security Project® (OWASP). In other words, the cloud native architecture is not actually solving any traditional software security problems, and these problems always need to be addressed.



4. Networking

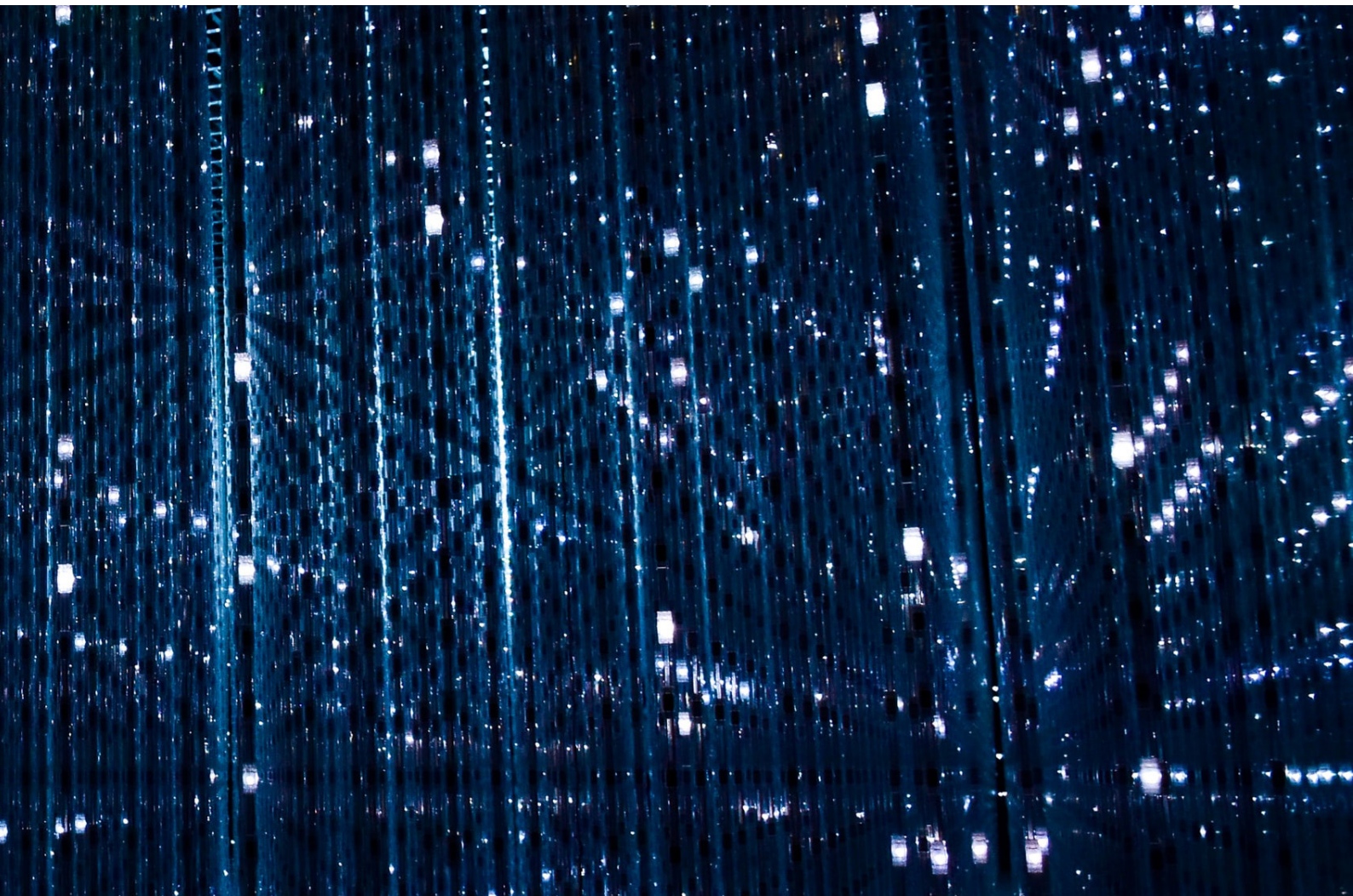
Containers communicate with each other and not all of them are expected to communicate with every other xxx. Compromised containers may be used to scan for other vulnerable containers and thus enabling an attack on them.

Kubernetes' network policies are the tool to help with that issue, allowing for the management of permissions for different pods as well as their ingress and egress traffic.

The old and proven technique of network segmentations is highly recommended to create network boundaries. In case of one compromised application, others won't be accessed thankfully due to the network separation.

On the other hand, it's worth considering to use a Service Mesh architecture which greatly helps with API security and traffic management. It's best to avoid the APIs from being called by any wrong callers in the first place. Throttling can help with the overloading of APIs, which is often a result of attempts to overload the API by an adversary.

It may seem obvious that everything is protected on the network layer by TLS, including internal components, but (again) it's not by default.



5. Monitoring

Monitoring of the communication and components activity is one of the aspects that can detect malicious activity. Pattern recognition techniques, including machine learning, can be of great help in detecting wrong behavior and reacting quickly to prevent further exposure of the app.



6. Automation to the rescue?

Yes and no. Of course it would be hard to deal with such complexity without automation, and in the era of everything as code, security automation is a strong trend. It applies especially to the detection part of finding misconfigurations and vulnerabilities, automated code reviews against security best practices, upgrading components, and shutting down compromised pods.

And, let's not forget about the tens or hundreds of tools offered by multiple vendors, all with the promise of making it easier to manage cloud native infrastructures and cloud native architectures; among these vendors are public cloud providers. We're all bombarded with product offers every day about another tool that is claiming to ease the complexity of the cloud native in a particular area, including security.

7. Security culture

A security culture is helpful, but it's not all that is required. The security of cloud native digital solutions requires a cultural change, which is being addressed by the **DevSecOps** movement and is supported by respective tool chains. Security is also a trade off and it's a determination made by humans of which rules to follow and how they will be implemented given the context of the risk profile of the solution and organization.

When a security culture is very low, security related tasks and processes are last on the priority list. Those actions even become facades for hiding both an **inability** and **unwillingness** to address the security of the cloud native solutions properly. This type of culture always ends in a **security disaster**. . . the question is just **when** and how big of a problem it is going to be. Building a solid security culture is much harder than applying a new security patch, so it's better to get started as early as possible.

8. If you think you're done...

The old mantra of security being a **process** and not a **project** applies to cloud native scenarios as well. New practices emerge, new tools with new capabilities, and even faster newer types of threats and exploits. Cloud native communities exchange their best practices for security as well as examples of common mistakes to avoid. So, it never ends and **there is no finish line** when we talk about security.



9. The future of secure cloud native solutions



In case anyone hoped it was going to be easier, it is not. Cloud native environments are complex ecosystems consisting of multiple services running together in different configurations. And, all of them are moving parts as they are actively developed by their respective open source communities. Older versions are deprecated, while new versions may come with new compatibility and security issues.

Some try to adopt a serverless paradigm that functions as a service running in the public cloud. Why? The cloud provider is responsible for maintaining the underlying security for the OS and containerization system, as well as the function runtime security and hardening. It takes away a lot of work from the internal team. However, the adoption of this paradigm will be slower than anticipated.

means a requirement for a combination of skill sets essential for both on-premises and multi-cloud (AWS, Azure, GCP, others).

The effort to maintain secure cloud native solutions is a necessity, not an option. It also means learning about new practices as well as rules to follow in order to make it as secure as possible. The flexibility which the cloud native delivers comes with great complexity, and that includes security challenges.

The golden rules of security, such as defence in depth, minimizing the attack surface, and zero trust have only gotten stronger and are not going anywhere.

10. Why Avenga ?



Flexibility, delivered by cloud native comes with complexity, including new security challenges. Avenga is here to help you apply modern, pragmatic approaches and deploy applications at scale, with a focus on cloud native infrastructure.

Avenga teams have a great deal of experience in both public cloud solutions and cloud native architectures in local environments. This combined knowledge is your ticket to accelerating your cloud transformation efforts with security as a first class citizen.



Contact us for more information

Jacek Chmiel is head of Avenga Labs, a division of Avenga focused on technology research and trend analysis as well as innovative technology-oriented proof of concept projects with business partners. He has played every role in the production process, overseen complex projects and successfully created concepts for highly sophisticated software and consulting services.

Jacek Chmiel

Director of Avenga Labs

Avenga

jacek.chmiel@avenga.com



New Jersey

Avenga USA
18 Overlook Ave, Suite 9
Rochelle Park
NJ 07662

www.avenga.com

Berlin

Avenga Germany GmbH
Köpenicker Straße 154
10997 Berlin

Warsaw

Avenga Poland
26 Przyokopowa Street,
01-208

