DEVOPSIFYING NETWORK SECURITY

An AlgoSec Technical Whitepaper

Introduction

This technical whitepaper presents and discusses the concept of “Connectivity as Code”, a complementary concept to “Infrastructure as Code” (IaC), and we will explain how it can be incorporated into the DevOps lifecycle for a more agile application delivery. We will also describe how empowering the developer to define the application’s connectivity requirements will bridge the gap between developers and network security, and help to automate the application delivery process end-to-end. The solution presented in this whitepaper seamlessly weaves network connectivity into the DevOps methodology, while ensuring continuous compliance, so that automation does not compromise security.

In addition, we will show that Connectivity as Code supports not only automation and agility in the application delivery process, but also bridges the gap between application developers and network security teams on an ongoing basis, even after the application is deployed into production. This is achieved by creating an abstraction layer that translates between the two worlds, giving application developers more control over their applications, while helping network security understand the business impact of their day-to-day tasks, and thus ensure business continuity.

The detailed technical descriptions in this document include real-life examples that can be used as is or can serve as a blueprint for implementing a DevOps process with automated network connectivity provisioning baked in.

The Network Security and DevOps Problem

DevOps is all about agility - fast, short delivery cycles - and automation. Enabled by recent technologies such as virtualization, cloud and SDN, spinning up new servers, provisioning storage in a public or private cloud or even spinning up whole environments now takes minutes or even seconds. Yet opening a port from a new server to a remote location on an internal or external network can take weeks, and there’s limited visibility into the process.

In a typical DevOps scenario, developers create a CI/CD pipeline which includes compiling and building processes, running unit tests, configuring test environments and running integration tests, and even pushing the new application into production – all completely automatically. But if the new application or version requires new network connectivity, the application developer needs to manually open a change request (out of band, e.g. using ServiceNow or other ITSM system), and then wait for approvals and implementation of the new connectivity flow before proceeding with the DevOps flow. This brings the entire process to a standstill, and negates the desired agility in the application delivery process.

Moreover, the developer is usually required to provide information about firewalls, zones, subnets, and other information related to the underlying network infrastructure in these change requests – information that is not always known or clear to the application developer. On the other hand, the application requirements and context are not always understood by the network security team assigned to implement the changes. This, in turn, results in back and forth between the developer and network security, making the entire process extremely long, error-prone, inefficient and frustrating to both teams.

The Solution: Connectivity as Code

Connectivity as Code closes this gap in the DevOps process. Connectivity as Code enables the application developer to describe the application connectivity requirements in a simple file, that lists the logical flows that represent these requirements. No additional details regarding the network infrastructure, firewalls, etc. is required. Once this list is created, a new phase in the DevOps process, called the Connectivity phase (detailed below), will automatically take care of things, with no further involvement required from the application developer. And whenever a new version of the application is developed, this list of connectivity requirements is updated by the developer, and the connectivity phase will ensure that the new flows are provisioned.

3.1 The "Connectivity as Code" File

The Connectivity as Code file[1] is a simple, machine-readable text file (e.g. structured as a JSON or YAML file), that represents the logical connectivity requirements of the application. It is presented as a list of abstract flows that typically include:

- Source (the consumer, who initiates the connection)
- Destination (the provider, who accepts the connection)
- Service
- Comments or additional information can be added as well

There is no need to know where the servers are located, what is the underlying network topology (e.g. whether there is a firewall between them, cloud security controls, etc.), or even their IP addresses or subnets. There is also no need to know whether connectivity is already available (e.g. because it's also required by other applications).

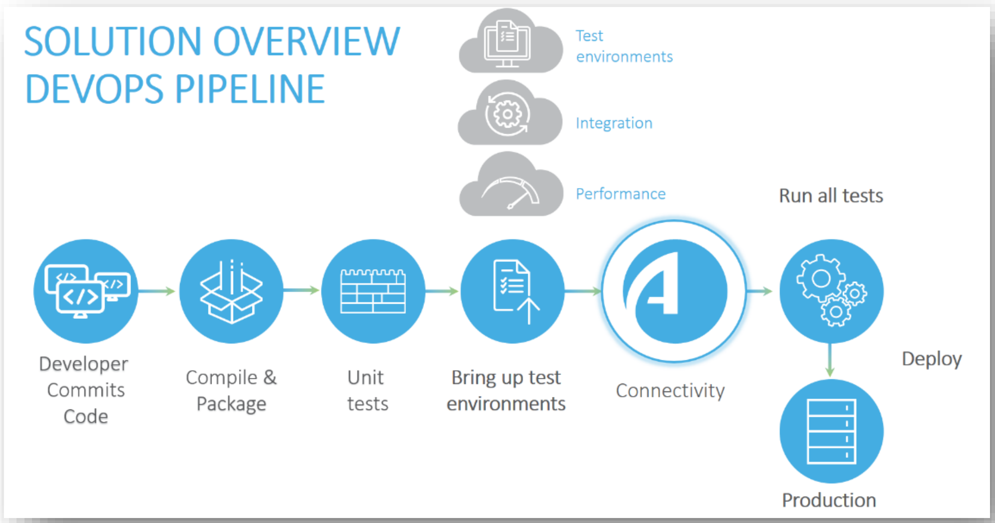
The file should include all connectivity requirements – between different components within the application (e.g. app server to DB), as well as external connections to/from the application (e.g. external clients connecting to the app web server). It should also include the connectivity requirements for the different application environments – dev, test, production, etc. – to ensure that connectivity is provisioned to allow both a smooth development and testing process, and to ensure that there are no surprises when eventually deploying the application into the production environment.

Once this file is created by the application developer, it is entered into the repository of files required for building the application, just like the code itself and any other supporting configuration files. The application's version control should also include this connectivity file for backup, easy rollback, application versioning and branching, etc.

[1] See appendix I for a sample "Connectivity as Code" JSON file, describing the connectivity requirements of an application.

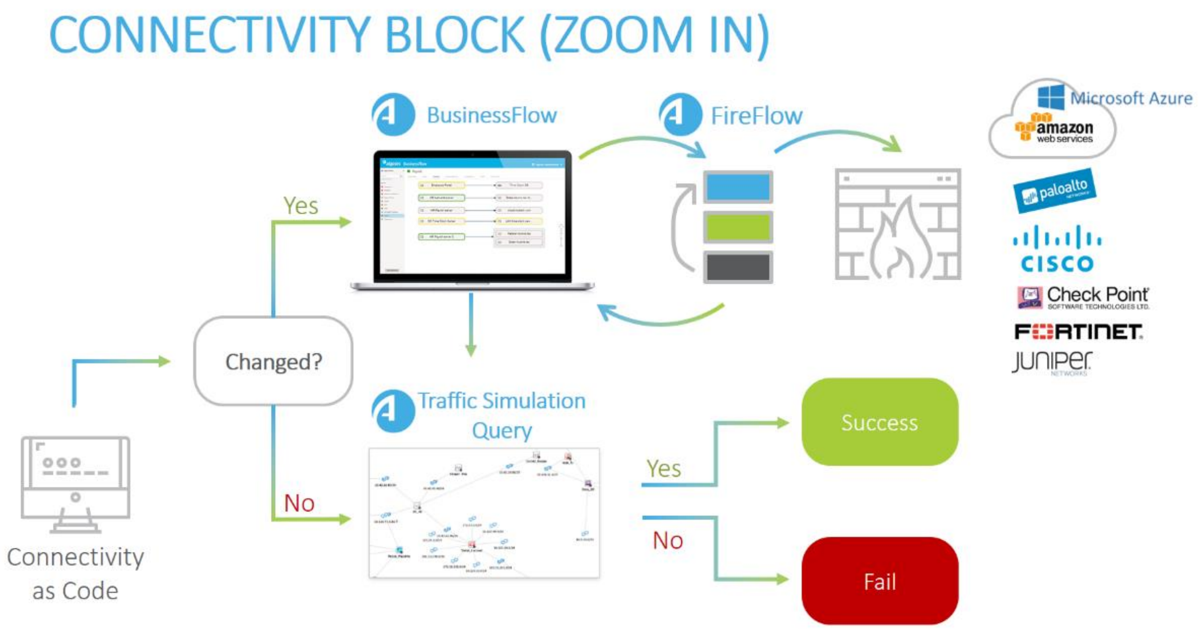
3.2 The DevOps CI/CD Pipeline

In order to automatically provision connectivity based on the Connectivity as Code file described above, a new phase is added to the CI/CD pipeline: The Connectivity phase.



The connectivity phase is where the actual connectivity provisioning and validation happens. This phase leverages AlgoSec's network security policy management capabilities to automate network connectivity provisioning as part of the DevOps process ("DevOpsify"), just like any other configuration or provisioning step in the DevOps pipeline.

In the Connectivity phase the AlgoSec solution follows this logic:



1. Read the Connectivity as Code file

2. Check whether the connectivity requirements have changed (new application, new flows required, some flows no longer needed, etc.).

   If no changes were made, AlgoSec will verify that the required connectivity remains in place, and that the application will work correctly. In this case nothing needs to be done – the connectivity phase has completed successfully and the CI/CD flow continues. This provides the application developer with the assurance that the necessary connectivity is in place and connectivity-related failures are not anticipated when moving into production.

   Note, this process relies on AlgoSec's traffic simulation and network analysis capabilities, which finds the firewalls/access-lists/filters/security groups that are in traffic path, and detects whether their security policy currently allows the requested connectivity or not.

   If the connectivity requirements have changed, AlgoSec's business application connectivity repository (AlgoSec BusinessFlow), will be automatically updated with the new connectivity requirements for this application. This information will be used for both provisioning the required connectivity now, and will also be retained for future reference, to ensure that the application's connectivity remains intact in the event of future network architecture changes, application or server migrations, or when additional filtering, policy cleanup, etc.

   If one or more of the required flows is currently blocked, AlgoSec will automatically trigger a security policy change request process. Through this process AlgoSec will verify that the new flows comply with the organization's pre-approved security policy, as well as security best practices and industry regulations, and will then design and implement the required changes directly on the different security devices on the network, automatically and within minutes.
   If, however, a change request is non-compliant, it will be escalated for approval and, once approved, it will be implemented automatically, thereby saving the developer the need to manually open an out-of-band change request.

   This change management step is performed by AlgoSec FireFlow, AlgoSec's change automation product, leveraging a zero-touch change workflow.

This concludes the connectivity phase, and the DevOps process will now continue onto the next phase.

Let's review a short example of this process:

Jane develops a new version of application Foo. This version includes a new spell-checking feature that requires connectivity to the SpellCheker service on the internet.

1. Jane updates the connectivity file and adds the following flow:
   "Foo AppServer" -> "SpellCheker Server" with service https
   Jane knows that by documenting the application's connectivity requirements, she will not only ensure a smooth deployment of the new version (even if some security policy changes are required), but that her application's connectivity will also be "future proof" (e.g. when a new firewall is introduced, or her application server is migrated to the public cloud).

2. Jane commits her changes and triggers the CI/CD process. After the code is compiled and packaged and all tests are successfully completed, the process continues to the connectivity phase.
3. The new connectivity requirement is detected, and the Foo application in AlgoSec BusinessFlow is automatically updated with this new flow.
4. BusinessFlow detects that this flow is currently blocked, and a change request is automatically triggered in AlgoSec FireFlow.
5. AlgoSec FireFlow detects that the perimeter Palo Alto Networks firewall is blocking the traffic.
6. AlgoSec FireFlow runs a Risk Check against the pre-approved policy created by information security, and detects that this traffic is approved.
7. AlgoSec FireFlow finds a rule in the perimeter firewall's policy that allows https traffic to several other internet services, and adds the SpellCheker server as a destination to that rule. It pushes the change to Palo Alto Networks Panorama and then commits and installs the policy onto the perimeter firewall.
8. The Connectivity phase is complete, and the process continues until the application is successfully deployed in production.

A couple of weeks later, Jane creates yet another version of Foo, but this time no new connectivity requirements are introduced.

1. When Jane commits her changes, the process runs again, and this time the connectivity phase quickly detects that no changes are required.
2. Connectivity is automatically verified by running a traffic simulation query to ensure that all relevant policies are configured correctly, and the process continues to another successful deployment into production.

How Does It Work

In order to implement the above solution, several key capabilities of the AlgoSec Security Policy Management Solution are used. In addition, AlgoSec provides several tools to easily integrate its solution with common configuration management and DevOps CI/CD solutions, as well as into home grown automation solutions and scripts.

The following sections describe the key AlgoSec capabilities that are utilized during the connectivity phase.

4.1 Network Model and Simulation

As part of its core technology, AlgoSec creates a network topology model that represents the entire organizational network. This model covers on-premise, private cloud and public cloud environments, and supports all the leading network security solutions, including firewalls, routers, load balancers and cloud security controls.

The network map model is then leveraged to perform traffic simulation queries, i.e. given a specific flow, what is the exact path it will traverse across the network, and whether any security devices along the path currently block the traffic.

This capability provides the basis for AlgoSec's ability to translate intent-based, abstract connectivity flows, into the technical requirements and specific network devices and policies in the underlying network security infrastructure. It is also used to design any changes required to allow missing connectivity.

4.2 Zero-touch Automated Change Workflow

When the connectivity phase detects that there are new connectivity requirements, a change request is automatically triggered in AlgoSec's automated change workflow solution, AlgoSec FireFlow.

The new change request will go through several key steps:

1. Initial Plan - finding the relevant firewalls and security devices that require a change.
2. Risk Check – verifying the new connectivity flow adheres to the pre-approved organizational security policy, as well as regulatory compliance requirements.
3. Work Order – designing the exact change that needs to be made on each security device, in the most optimal way for the specific vendor device and policy. This includes deciding which policies, zones, ACLs to update, reusing existing objects and rules to avoid clutter, enforcing naming conventions and documentation, etc.
4. ActiveChange – automatically pushing the designed change to the different security devices.
5. Validation – verifying the change was implemented successfully and connectivity is now available.

With AlgoSec, workflows, conditions and thresholds can be customized as needed (e.g. escalate to the security team's approval if a risk check detects change is about to raise a high security risk, affects specific firewalls, etc.).

In addition, AlgoSec provides full documentation and an audit trail which are built into the automated change process, keeping the security team in control and the auditors happy.

4.3 BusinessFlow - Business Application Repository

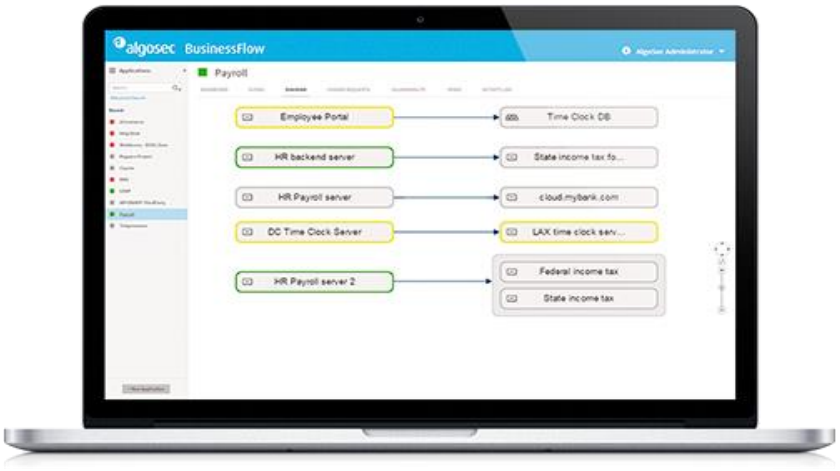
With Connectivity as Code, connectivity requirements of each business application are stored in AlgoSec BusinessFlow, and each change in the application's connectivity is automatically updated, so that it remains up to date.

This application repository can then be used either by the application developers or by the network security teams, as follows:

4.3.1 Application Developers/Owners

AlgoSec BusinessFlow gives application developers or owners visibility into the connectivity status, security, vulnerabilities and compliance, for their specific applications. It provides a clear summary of the application's connectivity flows (including a schematic diagram representing the connections within the application), changes over time, and additional information regarding the security and compliance posture of the application.

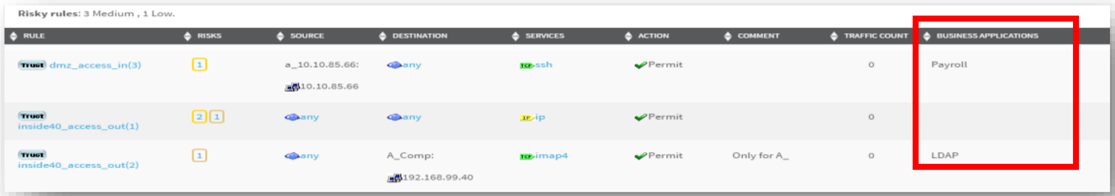
AlgoSec BusinessFlow can also be used as an interface to manage the application's connectivity requirements, and even as a mechanism to safely decommission a retired application without impacting any other active application.



Most importantly, AlgoSec BusinessFlow provides a very easy way to check the network connectivity status of the application, for example when troubleshooting an application in production.

4.3.2 Network Security and Architects, Security Operations

AlgoSec BusinessFlow automatically adds the business application context to every underlying network infrastructure device, object or rule - i.e. each firewall rule is automatically labeled with the business applications it supports. A quick analysis can then be performed to check the potential impact of any security change or maintenance activity, such as isolating a compromised server, taking down a server, firewall or router for maintenance or upgrade, cleanup of firewall rules, network architecture changes, etc., on business continuity. This capability alleviates the need to attempt to reverse engineer a firewall rule to find its associated applications, and enables safe and efficient management of the network.



Risky rules: 3 Medium , 1 Low.

RULE	RISKS	SOURCE	DESTINATION	SERVICES	ACTION	COMMENT	TRAFFIC COUNT	BUSINESS APPLICATIONS
Trust dmz_access_in(3)	1	a_10.10.85.66: 10.10.85.66	any	TCP ssh	✓Permit		0	Payroll
Trust inside40_access_out(1)	2 1	any	any	IP ip	✓Permit		0	
Trust inside40_access_out(2)	1	any	A_Comp: 192.168.99.40	TCP imap4	✓Permit	Only for A_	0	LDAP

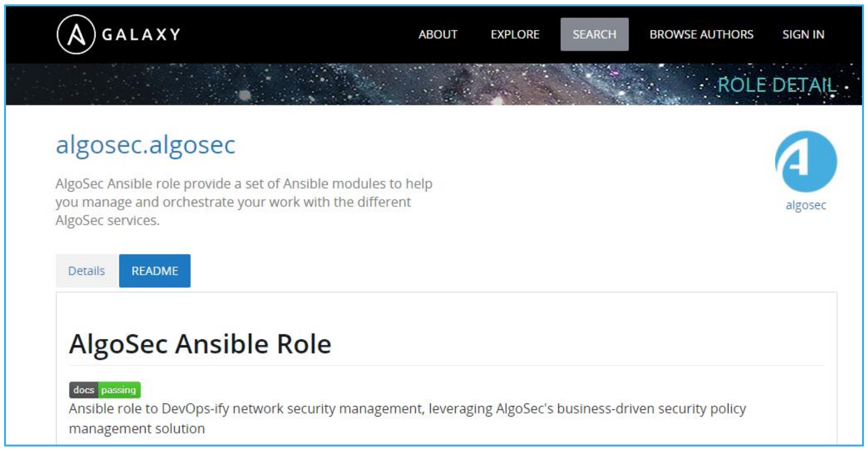
4.4 Integration with Orchestration and Configuration Management Flows

In order to integrate Connectivity as Code into commercial or home-grown CI/CD orchestration solutions, AlgoSec provides several tools and sample integration code.

4.4.1 Ansible

If using Ansible as the DevOps orchestrations and configuration management system, adding the connectivity phase to the DevOps process is as simple as downloading the AlgoSec Ansible role, creating a suitable playbook (sample available), and plugging the playbook into the existing flow.

The AlgoSec Ansible role, sample playbook and a sample Connectivity as Code inventory file, as well as comprehensive documentation, are available for download from the Ansible Galaxy:



algosec.algosec

AlgoSec Ansible role provide a set of Ansible modules to help you manage and orchestrate your work with the different AlgoSec services.

AlgoSec Ansible Role

Ansible role to DevOps-ify network security management, leveraging AlgoSec's business-driven security policy management solution

The AlgoSec Ansible role is distributed as open-source, and can be used as is or customized as needed.

The AlgoSec Ansible role is implemented in Python, using the AlgoSec Python SDK and the AlgoSec APIs (see hereafter), and connects to the AlgoSec server deployed in the organization to perform the necessary queries and actions.

Similar modules will be available for other common orchestration solutions such as Chef, Puppet and Jenkins, or can be created as needed, using the Ansible role as a reference.

4.4.2 Python SDK

AlgoSec also distributes a Python SDK, which serves as a Python wrapper for AlgoSec's APIs. It implements the logic required to implement the Connectivity as Code phase in the DevOps flow, as described above.

The AlgoSec Python SDK is distributed as open-source and available for download from GitHub.

4.4.3 APIs

Alternatively, AlgoSec's rich API set can be used with any programming language and any environment, as needed.

See the AlgoSec API guide for more details.

Summary

This whitepaper described a process to implement the Connectivity as Code approach in order to weave network connectivity and security into the DevOps process, or "DevOpsify" network security.

By incorporating Connectivity as Code into the DevOps process, organizations will benefit from:

- Seamless management of network connectivity as part of the DevOps process for faster, more agile and problem-free application delivery, rather than as an external out-of-band issue that requires separate - and manual – handling.
- Continuous compliance and auditability throughout the application delivery process.
- Business continuity – application connectivity requirements are clearly documented and up to date, ensuring minimal disruption to the business even during network, infrastructure or architecture changes.
- The ability to bridge the gap between application developers and network security throughout the entire application lifecycle (planning and development, deployment, production and decommissioning)

About AlgoSec

Appendix I - Sample Connectivity Repository File

This json file lists several connectivity requirements (or flows) for a ‘Billing’ application, and one flow for a ‘Payroll’ application.

Note: Users can choose to create Connectivity as Code files for each application separately, or group the connectivity requirements of several applications into one file (clearly listing which flows are needed to support which application).

```
{
  "applications": [
    {
      "app_name": "Billing",
      "app_flows": {
        "flow1": {
          "sources": ["HR Payroll
server", "192.168.0.0/16"],
          "destinations":
["16.47.71.62"],
          "services": ["HTTPS"]
        },
        "flow2": {
          "sources": ["10.0.0.1"],
          "destinations":
["10.0.0.2"],
          "services": ["udp/501"]
        },
        "flow3": {
          "sources": ["1.2.3.4"],
          "destinations":
["3.4.5.6"],
          "services": ["SSH"]
        }
      }
    },
    {
      "app_name": "Payroll",
      "app_flows": {
        "new-flow": {
          "sources": ["1.2.3.4"],
          "destinations":
["3.4.5.6"],
          "services": ["SSH"]
        }
      }
    }
  ]
}
```