DOI:10.1145/3639702

Intent-based networking (IBN) offers advantages and opportunities compared with SDN, but IBN also poses new and unique security challenges that must be overcome.

BY JIWON KIM, HAMED OKHRAVI, DAVE (JING) TIAN, AND BENJAMIN E. UJCICH

Security Challenges of Intent-Based Networking

COMPUTER SYSTEMS HAVE evolved over decades to enable more flexible programmability. Unsurprisingly, such programmability has converged more closely to how humans think and speak. This is perhaps best exemplified in the evolution of programming languages: an obtuse set of ones and zeros (machine language), small snippets of instructions (assembly language), early low-level abstractions with human-readable syntax (for example, ALGOL or C), and today's high-level abstractions with complex operations (for example, Python or Go). With each subsequent generation, programming languages have described richer semantics that enables desired functionality to be expressed more naturally. Not only has this evolution made programming easier, but such evolution allows humans to focus on *what* they want to do instead of *how* the machine should be programmed to do it. In other words, the evolution of high-level semantics has allowed humans to focus on the *intent* instead of the *mechanism*.

A similar evolution has occurred in computer networking. Early computer networks were statically configured machines that required manual configuration. Over time, network semantics evolved into distributed, self-configuring switches and routers. But, with the complexity of modern networks (for example, large enterprises, data centers), such a model became unscalable and unsustainable because network operators had to deal with individual rules on switches to configure and debug a network.

In recent years, software-defined networking (SDN) was developed as an answer to this need by capturing richer

» key insights

- **Intent-based networking (IBN) allows network operators to more easily define what they want their network to achieve through high-level intents rather than worrying about complex, low-level configuration details of how the network achieves it.**
- **The separation between high-level intents and low-level configuration details creates a semantic gap between intended network objectives and actual behavior, which introduces security (and privacy) challenges that threaten the correct operation of the network.**
- **Systematically studying the threats facing IBN uncovered several new attack vectors not found in legacy networks or traditional SDN networks.**
- **Given the nascent state of IBN adoption by industry and vendors, we propose several opportunities and directions for IBN security research going forward.**

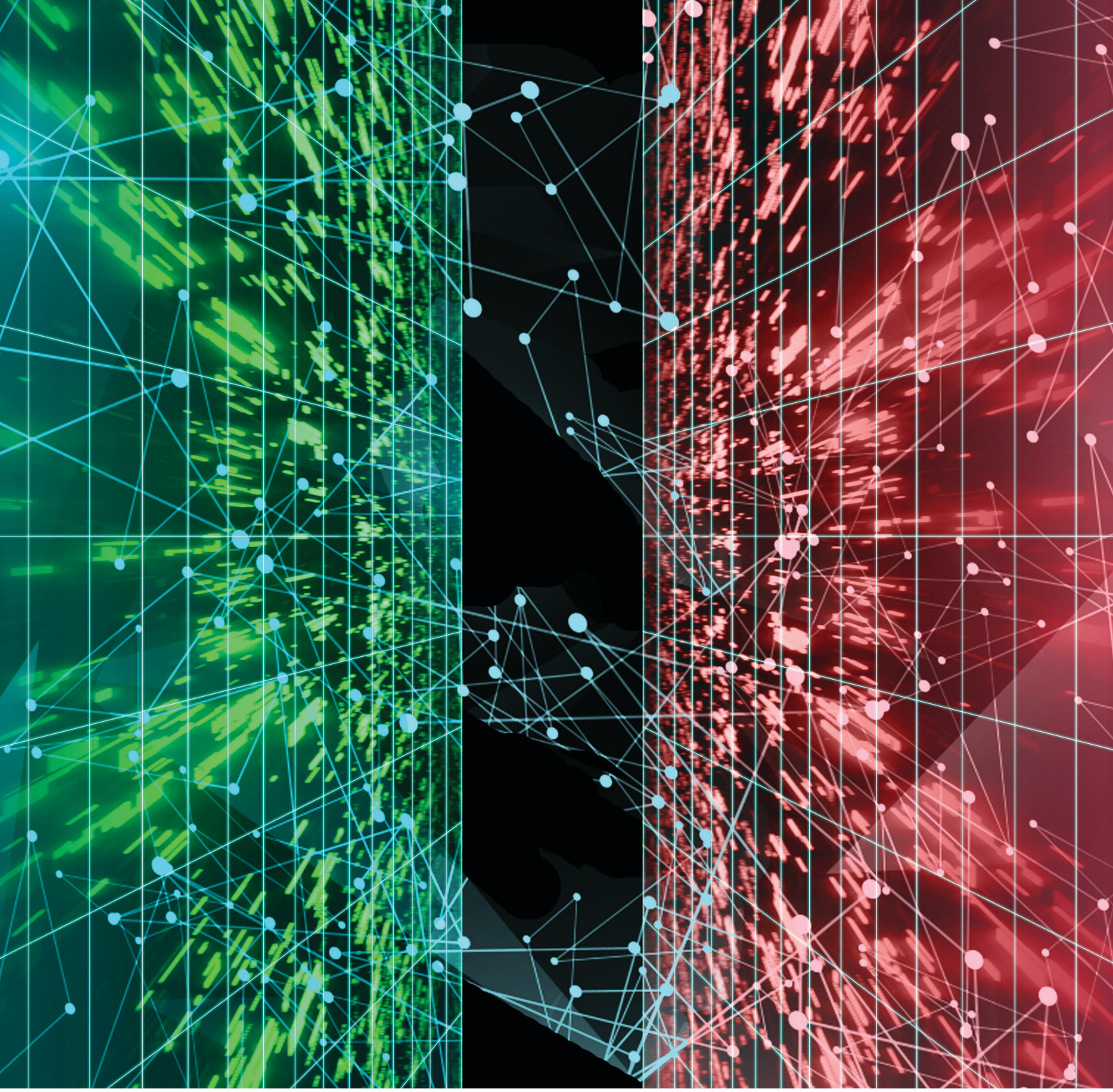


IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK

semantics about network operations. SDN separates the control plane (that is, the decisions about traffic flow) from the data plane (that is, the traffic itself),[23] with an SDN controller that acts as a central vantage point. However, such separation still requires network operators to understand their networks' low-level protocols and policy rules while simultaneously dealing with increased complexity, diverse environments, and scalable capability demands.

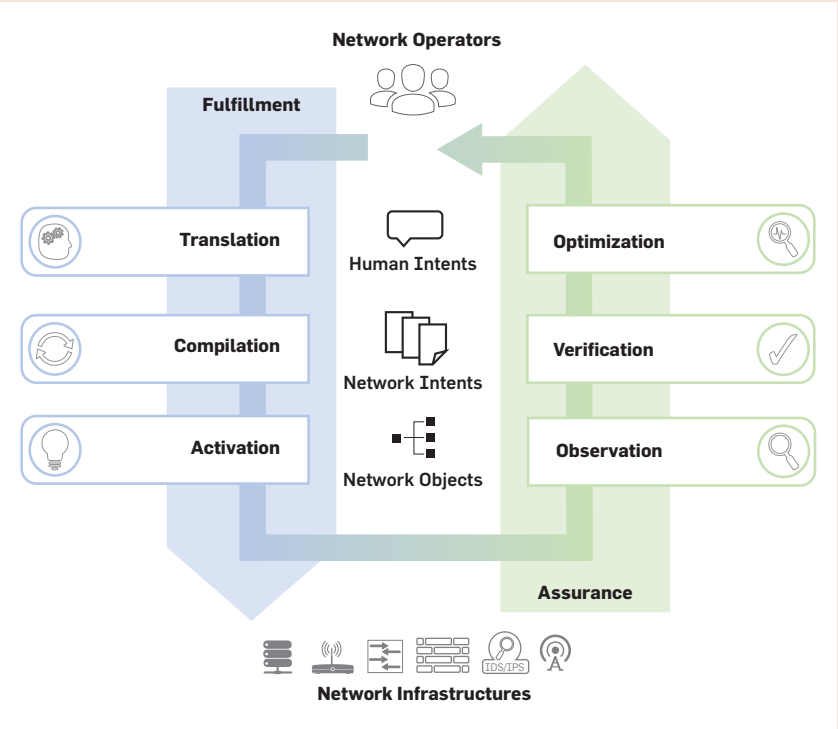
Presently, the next progression in network evolution comes from *intent-based networking* (*IBN*). Informally, IBN allows a network operator (that is, network administrator or network programmer) to describe their high-level *intent* for what they want to happen in the network as opposed to how such actions should be achieved with individual traffic rules or particular protocols. Such an approach is analogous to a higher-level programming language that represents a richer set of operations and instructions.

However, just as with programming languages that employ high-level abstractions, such abstraction comes at the cost of further widening the semantic gap between the interface provided to the operator and the actual traffic-forwarding rules and protocols used in the network. Unfortunately (and critically), the potential security and privacy implications of this semantic gap have not been explored systematically to date.

In this article, we provide a brief overview of the advantages and opportunities that IBN provides. Based on those characteristics, we then

Figure 1. Intent lifecycle within IBN.

The lifecycle includes two main stages, *fulfillment* and *assurance*, that ultimately form a feedback loop toward automated, semantically rich network functionality.



systematically analyze the unique security challenges that IBN poses and propose several opportunities to tackle these challenges. Given the recent interest from industry and academia in IBN, we posit that a thorough understanding of the IBN security posture will foster greater interest and deployment in this latest evolution of networks.

Background

IBN is best contrasted with traditional networks by its ability to allow network operators to define *what* the network should do instead of *how* it should be done. While SDN reduces the burden of manual configuration efforts, network operators must still build and develop custom applications that require knowledge about lower-level network events (for example, network forwarding rules, configuration protocol limitations, and so on). In such environments, operators without domain knowledge could construct networks incorrectly.

IBN reduces this burden by abstracting away network implementation details, such as data-plane forwarding rules, network configuration protocols (for example, OpenFlow, SNMP, OSPF, BGP), and the underlying physical (or virtual) topology.

IBN has been standardized by the Open Networking Foundation (ONF),[27] the Internet Research Task Force (IRTF),[12] and the 3rd Generation Partnership Project (3GPP).[1] Although differences in terminology exist among the standards, all of them have a common model in which a centralized *intent controller* (or *IBN controller*) manages diverse types of intents based on an *intent lifecycle*. The intent controller may be realized as a subsystem of a network operating system (NOS) or an SDN controller.

Within the open source community, Open Network Automation Platform (ONAP), Open Network Operating System (ONOS), and OpenDaylight (ODL) provide high-level intent interfaces for network operators. From industry, multiple commercial products officially support IBN, including Cisco IBN[10] and Juniper Apstra.[26] Google has also announced its internal use of IBN in its latest SDN-based environment.[14]

Intent properties. IBN's key feature lies in the *intent* of what the network should do. An operator specifies a user-defined intent as a declarative set of general *properties* or *constraints* that must be met. Previous efforts[3,29,31] have proposed diverse properties that can be incorporated into intents, and extending properties for expressive policies is an active research area.

For instance, some basic network properties can be expressed as constraints within intents:

- *Reachability* guarantees connectivity between nodes in the network (for example, "allow a host with an IP address of 10.0.0.1 to communicate with a host that has an IP address of 10.0.0.2").
- *Waypointing* specifies partial or complete paths for routing (for example, "traffic destined to the database server should pass through the firewall first").
- *Bandwidth* allocates the data-plane bandwidth for a particular purpose (for example, "traffic destined to the database server should be at most 500 Mbps").
- *Isolation* enforces that traffic is physically or virtually isolated from other traffic (for example, "traffic for disaster recovery must use dedicated and redundant network slices to avoid creating a single point of failure in the network").

Intents can also be used to group similar devices together (for example, all hosts within a department of an organization) and can join together multiple constraints. Intents may include *stateful* constraints (for example, "the firewall must mirror traffic with more than five failed connections to a monitoring server") or *temporal* constraints (for example, "traffic from the marketing team is only allowed from 9 a.m. to 5 p.m., Monday to Friday").

In short, intent properties enhance the expressiveness of how network behavior can be declared without placing the focus on lower-level details (for example, network topology or network configuration protocol). This expressiveness also enables the reduction of complexity for network operators who no longer have to reason about behavior solely through low-level configuration changes.

Intent lifecycle. Once a network operator has declaratively specified an

intent and any associated intent properties, the operator submits the intent to the IBN controller. The intent follows an *intent lifecycle*, as shown at a high level in Figure 1.

The intent lifecycle consists of two phases: *fulfillment* and *assurance*. This is analogous to programming languages that have compilation and debugging phases. The fulfillment phase transforms intents from operators into a set of network objects applied to network infrastructures. The assurance phase confirms the correctness of intents by reading network behaviors from devices and re-interpreting them as high-level intents. During the two phases, the IBN controller continuously reconciles the inconsistencies between the intents and the network; verifies network invariants, such as the absence of loops or blackholes; and validates the high-level intent semantics. Since an intent exists in a specific lifecycle stage at any given time, IBN can represent each stage as an *intent state*.

Throughout the remainder of this article, we use a demonstrative example of a marketing team that requests network capabilities within an enterprise organization's network. Figure 2 shows the corresponding abstractions and events in the intent lifecycle.

Intent fulfillment. The fulfillment phase translates high-level intents to low-level network objects following three stages: translation, compilation, and activation.

Translation. Translation converts user-defined intents into network-level intents. Translation allows non-expert users to simply define their desired goals using a human language or a declarative language, and IBN can leverage natural-language processing (NLP) or a domain-specific compiler to accomplish this task.

Compilation. Compilation transforms network-level intents into network objects, which include abstractions such as device configuration information, forwarding rules in the data plane, and network topology. To bridge the gap between users and network behaviors, the IBN controller uses the network-level intents and a global network view to calculate viable forwarding rules that could satisfy the requirements. For example, in Figure 2, the IBN controller calculates all paths between the marketing team's hosts and the database hosts by referring to group information and the underlying network topology.

Activation. Activation implements and realizes the network objects in the underlying network devices. Activation can be delegated to the existing responsible system, such as an SDN controller.

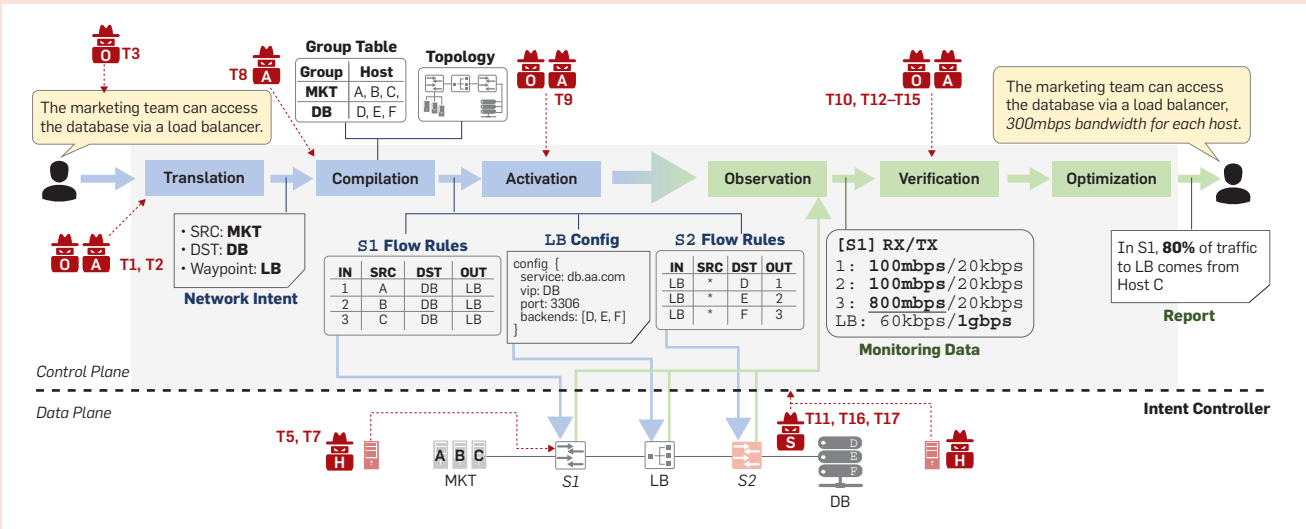
In Figure 2, the activation stage controls network devices by sending flow (forwarding) rules on software switches (for example, switches *S*1 and *S*2), changing configurations of devices (for example, the load balancer *LB*), and executing remote commands. Activation may also modify the executable code present on programmable devices (for example, a programmable data-plane configuration implemented by the data plane programming language P4). For any errors, such as a flow table overflow, the IBN controller will return the error message to the compilation stage to recompile an intent from which the object originated.

Intent assurance. Once an intent has been fulfilled, IBN continues with a goal of continuous assurances that the network performs as expected. Such assurances enable greater insight capabilities to proactively optimize and suggest network improvements to the operator. The assurance phase consists of three stages: observation, verification, and optimization.

Observation. Observation enables the construction of a centralized and global view of the network. Such observation occurs by periodically monitoring the network and synchronously receiving any messages from network devices. One goal from observation is to measure the network's performance

Figure 2. IBN workflow and adversarial threats.

A network operator (that is, a user) specifies a high-level network intent that is fulfilled through low-level configuration in the data plane. IBN employs network observation to enhance verification, optimization, and recommendation capabilities. Attacks from network operators (O), applications (A), switches (S), and hosts (H) can exploit threats in Table 1.



to detect possible degradations. For instance, a switch in Figure 2 might experience a traffic bottleneck; depending on configuration, the switch may proactively notify the IBN controller or the IBN controller may reactively detect such traffic conditions.

Verification. Verification confirms whether given intents are correctly implemented. If the IBN controller determines that an intent is not activated correctly after observation, the intent can be re-compiled with new parameters.

For instance, if there are two intents that compile down to flow rules that use the same matching fields (for example, a TCP/IP 5-tuple) but that implement different actions (for example, forward traffic on a particular port) on the same switch, the traffic may match successfully with the low-priority intent before such a match is checked against the high-priority intent. That distorts the goal of the high-priority intent taking the highest priority.

To avoid such situations, the IBN controller can choose a new path for the low-priority intent. Additionally, whenever the network state changes (for example, link failure), the verification process can help select affected intents for re-compilation and assess the effectiveness of given intents using the network statistics gathered during observation.

Optimization. Optimization summarizes network behavior and recommends suggested improvements. When an issue occurs in traditional networks, operators attempt to understand the root causes of the issue using a set of primitive tools (for example, `ping`, `traceroute`, and `iperf`) and check that the individual device configurations are implemented as expected. Such an approach does not scale well for large networks and is subject to human error and misinterpretation.[a]

IBN summarizes the network's current status and reports key findings, which reduces the manual effort required to diagnose or improve network configuration. For instance, in Figure 2, the IBN controller reports to a network operator that 80% of the traffic destined to the load balancer *LB* via switch *S*1 comes from host *C*. Optimization can also automatically produce recommendations and subsequently implement them. An IBN-based network can learn from such summaries and suggest new intents. For instance, in Figure 2, the IBN controller suggests a new 300Mbps bandwidth constraint that the network operator can implement or that the IBN

a According to Cisco, up to 95% of network updates are manual, which increases operational costs two to three times.[11]

Table 1. IBN threat model overview. Attacks can originate from network operators (O), applications (A), switches (S), or end user hosts (H).

Stage	Challenge	Security Threat	Security Impact	Attacker	Example(s)
Translation/ Compilation	Intent Compiler Trust	T1 Installing incorrectly compiled intent	Intent Integrity Violation / DoS	O, A	CVE-2021-38363
	Intent Composability	T2 Installing overlapping intent	Intent Integrity Violation	O, A	CVE-2022-29944
		T3 Leaking unauthorized intent	Information Disclosure	O	
	Intent Linearity	T4 Installing intent that causes a side intent	Covert Traffic / Unauthorized Access	O, A	
		T5 Exploiting existing side intent	Covert Traffic / Unauthorized Access	H	
	Intent Granularity	T6 Installing coarse intent	Covert Traffic / Unauthorized Access	O, A	
		T7 Exploiting existing coarse intent	Covert Traffic / Unauthorized Access	H	
Compilation	App-Intent Interactions	T8 Installing intent from an unprivileged app	Elevation of Privilege	A	
Activation	Activation Conflicts	T9 Installing intent that causes object conflict	Intent Integrity Violation	O, A	CVE-2021-38364
Verification	Intent-Forwarding Rule Consistency	T10 Installing intent unsupported by network	Intent Integrity Violation	O, A	CVE-2022-29605
		T11 Exploiting verification vulnerability	Intent Integrity Violation / DoS	H, (S)	CVE-2022-24035
	Intent State Correctness	T12 Installing intent that causes inconsistent state	Intent Integrity Violation	O, A	CVE-2022-29607 CVE-2022-29609
		T13 Installing intent that corrupts state	Intent Integrity Violation	O, A	CVE-2022-29604 CVE-2022-29606
	Network Degeneracy	T14 Installing intent that violates network invariant	Intent Integrity Violation	O, A	CVE-2022-29608
		T15 Installing non-functional intent	Resource Exhaustion (DoS)	O, A	
Observation/ Optimization	Optimization Trustworthiness	T16 Exploiting optimization vulnerability	Unauthorized Access	H, (S)	
		T17 Sending fake monitoring or telemetry data	Covert Traffic / Unauthorized Access	H, S	

controller implements automatically on the operator's behalf. The overarching goal of optimization is to reduce operational costs and improve the network's utilization, reliability, and security.

Security Challenges

One market survey[35] estimates that the IBN market had a $1.27 billion valuation in 2021 that will eventually reach $5.09 billion by 2026. As organizations consider deploying IBN to enhance their network's functionality and to reduce their operational costs, they must also weigh IBN's security posture in the adoption calculation. Without a clearer understanding of IBN's security challenges (and opportunities), organizations may be hesitant to adopt IBN in spite of its operational benefits.

To date, no systematization exists on the unique and inherent security challenges within IBN. Our goal in this section is to highlight such security challenges and categorize them according to the two lifecycle stages of *intent fulfillment* and *intent assurance*. We note that all the challenges are, in some way, attributed to the large semantic gap between an operator's network view and the low-level implementation in network devices.

Threat model. We consider a threat model based on various actors in IBN, including network operators, network applications, network forwarding devices (for example, switches, routers), and hosts (that is, traffic generated from end users). Figure 2 shows an overview of the attackers and the attack surface.

As shown in Figure 2, even if the intent controller itself is considered trusted, attackers can still attack almost all stages in IBN with negative consequences. From the control plane, attackers may be network-operator insiders or malicious network tenants. From the data plane, attackers on the network's hosts can send arbitrary packets into the network. Such attackers may be able to indirectly exploit controller configuration, attack existing network devices with known vulnerabilities, or deploy malicious network functions that interact with the intent controller.

Table 1 provides an overview of 17 threats, **T1–T17**, that are explained throughout the rest of this section. We highlight the fact that about half of these threats have been shown to be exploitable in production-quality implementations, which are marked with relevant example CVE identifiers.

Attacking intent fulfillment. Given that fulfillment introduces a new abstraction layer with new input formats to operators and users, the abstraction can bring unique challenges related to intent translation, compilation, and activation that do not exist in traditional networks. We highlight several ways attackers can misuse these mechanisms.

Attacking intent compiler trust. Faithfully translating intents into forwarding rules crucially depends on a unique component in IBNs: the compiler. Software bugs in the compiler can cause not only syntactic errors that are usually visible to users with a system failure, but also semantic errors that often do not cause any crash and instead result in violating the integrity of how the network should operate (T1).

Semantic errors in IBN can cause two types of misbehaviors that attackers can exploit: a desired intent is not implemented in the network, or an undesired intent is implemented. To resolve the former case, the verification stage should validate the installed intents. However, since there is no general sound and complete algorithm for the verification problem, practical verifiers often focus on soundness, which means they are incomplete.

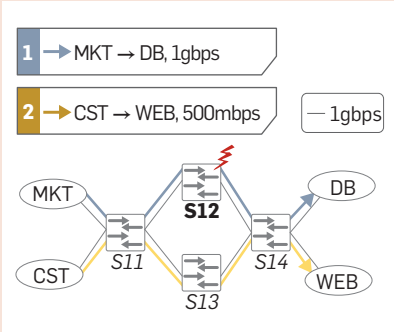
Attacking intent composability. Intents can partially or fully overlap with one another with respect to their reachability, bandwidth, and temporal properties (T2). As a result, IBN may be able to fulfill multiple intents when considered in isolation, but the network may not be able to meet the intended properties when combined. That could result in a denial of service attack against network availability.

Figure 3 shows an example that violates intent composability. While an alternative path exists for either intent to use S_{13} in the route between both sides of the network topology, the IBN controller selects the same path for both intents. That causes a bottleneck in S_{12} that could have been avoided with proper composability.

Conflicts among intents can be exceedingly hard to avoid. Intent APIs are often quite expressive and sometimes leverage a natural language. While composition tools[3,29,31] can synthesize specific types of intents, they require domain-limited, structured inputs such as graphs or regular expressions. If users' intents cannot be represented in such formats, IBNs cannot apply these tools to resolve their conflicts.

Figure 3. Violation of intent composability.

Intents 1 and 2 can independently meet their desired behaviors but may cause denial of service if the global view of all intents' constraints is not considered.



Complicating this matter are dynamic functions in translation or compilation stages that evolve over time. Unlike predefined types of intents, it might be infeasible to even define a conflict for intents with new conditions and parameters created by these functions.

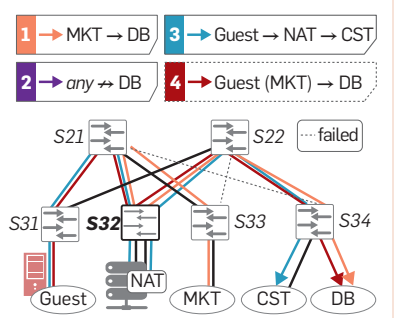
Finally, the multi-tenant nature of IBN makes composition a challenging security problem because composition conflicts and vulnerabilities can arise among different tenants' priorities. For instance, a malicious tenant may infer information about network utilization or other tenants' intents depending on carefully crafted intents that the malicious tenant attempts to install (T3).

Attacking intent linearity. New intents may also trigger "side intents" when combined with existing intents (T4). A *side intent* is traffic that is implicitly allowed but does not exist in any of the intents individually. Side intents allow attackers to escalate their access privileges to violate availability policies.

Figure 4 shows a violation of intent linearity. The first and the second intents try to permit only the marketing team (MKT) to connect to the database (DB). The third intent tries to allow guests to access the customer service team (CST) via a NAT middleware box (NAT). Data-plane attackers can launch link-flooding attacks[18] to congest targeted links such as (S_{22}, S_{33}) and (S_{21}, S_{34}). When the (S_{22}, S_{33}) link fails, traffic

Figure 4. Violation of intent linearity.

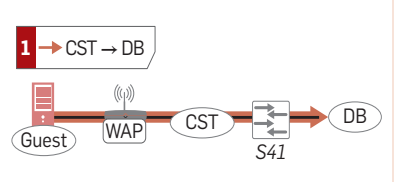
Switch S32 implements three intents that collectively enforce the policies that only MKT can connect to DB and that guests accessing CST must go through NAT first. One result is that a side intent, S, implicitly allows traffic from guests posing as MKT to access DB.



#	IN	SRC	DST	OUT
1	S21	MKT	DB	S22
2	*	any	DB	drop
3	NAT	NATed	CST	S22
3	S21	any	CST	NAT

Figure 5. Violation of least granularity of an intent.

An operator may wish to allow only representatives of a customer service department (CST) to access a database (DB), but how the intent is specified may unintentionally allow extra traffic to access DB.



from MKT to DB must go through S_{32}. Similarly, when the (S_{21}, S_{34}) link fails, traffic from guests to CST must also go through S_{32}. Finally, the guest attackers can leverage the two intents' behaviors to access DB by sending a crafted packet that spoofs MKT as the source (T5). Since the implicit side intent that is exploited (that is, connectivity between S_{31} and DB) is not explicitly visible in the flow table, it is difficult for network operators to detect such conditions.

Attacking intent granularity. One of the classic principles of secure design is the principle of least privileges.[30] For IBN, this requires the finest granularity of intent possible to avoid allowing greater connectivity than desired. A *coarse intent* can affect the network in unexpected ways and allow undesirable communication (T6). Coarse intents can conflict with each other because they cover larger subsets of the traffic space. Coarse intents can arise either from operators implementing an under-specified (that is, over-provisioned) privilege policy or from the IBN controller incorrectly implementing the network's compilation and activation mechanisms.

Figure 5 shows an example of an unintended consequence of a coarse intent that over-provisions privileges. If the network operator simply defines the intent to provide connectivity between the customer service team (CST) and a database (DB), any guest of the CST department may gain unauthorized access to the database (T7).

Exploiting application–intent interactions. Modern network operating systems that use SDN allow for external network applications to extend the control plane's functionality. In SDN, applications subscribe to events of interest so that the applications can consume and react to the events. For example, a Layer 2 forwarding switch application may subscribe to new packets coming from the data plane that have not matched existing forwarding rules, and the application may choose to install new forwarding rules in the data plane.

Similarly, network applications can subscribe to intent-related events via the IBN controller. That enables external network applications to install, reconfigure, remove, or observe intents. In such an environment, it is critically important to the overall network security posture for operators to be able to define the privilege model between intent applications and the IBN-enabled network.

The effects of granting privileges may not be obvious. For instance, an attacker that does not have permission to request intents may indirectly install intents by invoking cross-plane[34] network events to which the intent applications subscribe (T8). This leverages triggered events in the complex interdependencies among network objects and the event-driven nature of network operating systems.

Attacking activation conflicts. Intents with different objectives can have conflicts in implementation. This can occur when the compiled forwarding rules overlap, or when competing network configurations in network devices cause unintended consequences (T9). When a new intent with a higher priority has a conflict with an existing intent, the IBN controller should recompile the existing intent and try to activate it in other ways. If there is no way to reconcile these conflicts, the IBN controller should notify operators as to the unfulfilled intents. Otherwise, allowing activation conflicts will result in violating integrity of existing intents.

IBN controllers may be exploited if they handle multiple intents with identical network objects. In such cases, deleting one of the intents or changing the network configuration may cause unwanted behavior. For example, in CVE-2021-38364, deleting one of two intents that shared the same flow rules violated the other intent's integrity.

Attacking intent assurance. Intent assurance validates the various layers of abstraction in IBN, from high-level intents to low-level network objects. Assurance monitors all network events to perform process verification, summarization, and recommendation. However, the semantic differences between user intents and those intents' low-level implementation in the network create unique challenges. We highlight several ways that attackers can manipulate assurances to violate the integrity and availability properties of the network.

Attacking intent–forwarding rule consistency. Intents should be consistent with underlying network specifications, such as device configurations, any supported protocols, and the network's topology. Given that IBN abstracts away such details from users, users may unintentionally request intents that are beyond the capabilities of the underlying network (T10).

Improper handling of unsupported intents by the IBN controller may cause adverse effects on data-plane behavior and control-plane processes. For example, in CVE-2022-29605, an IBN controller repeatedly attempted to install an intent's IPv6 forwarding rules into legacy switches that did not support IPv6, which misleads a network operator by showing an in-progress state.

Moreover, intents should be consistent across network changes. For in-

stance, in CVE-2022-24035, an installed intent did not respond properly to subsequent network changes, which could result in a denial-of-service attack when combining link-flooding attacks (T11). Such inconsistencies can be hard to resolve for non-specialists, given that the network's details are more opaque to operators.

Attacking intent state correctness. An intent's state should correctly represent the current stage at a specific time and the outcome of the intent. However, if the intent state is inconsistent with the intent processing, it will mislead a network operator or the intent controller, which can be used to attack control plane processes (T12). For instance, in CVE-2022-29607, modifying an existing intent to have the same source and destination address showed that the intent was installed without any flow rule, even though it should have been a failure.

If an intent that is expected to be correctly installed shows an internal corrupted error state, adversaries can exploit this to attack IBN (T13). For example, in CVE-2022-29606, an intent could be corrupted with a large switch port number, which allowed an attacker to bypass any intent by assigning the large number to the host port.

Exploiting network degeneracy. IBN should consistently verify network invariants, such as the absence of forwarding loops and black holes. While a large body of literature in network verification has focused on these invariants, IBN faces additional challenges because operators (and developers) must find causal relationships between intents and conditions that cause the network to degenerate into undesirable behavior and violate network correctness properties (T14). For instance, in CVE-2022-29608, a one-way intent generated a forwarding loop in certain network topologies, which made it difficult to track the origin.

Another set of network degeneracy conditions can be caused by invalid intents, such as a zero-bandwidth intent. Without proper logic to detect such intents, the IBN controller could spend unnecessary resources to process invalid intents or even install ineffectual network objects, which ultimately affects control-plane and data-plane performance (T15). For example, in CVE-2022-29607, one controller implementation allowed an intent with the same source and destination addresses to be installed.

Exploiting optimization trust. While optimization can be useful for efficiently using all the resources of a network, undermining the information used in optimization can create security challenges. Optimization algorithms and their implementations must be correct and trustworthy to avoid introducing vulnerabilities. Attackers can exploit such vulnerabilities to influence network decision making or to achieve undesirable network conditions (T16).

In addition to the optimization code, the monitoring and telemetry data being observed must also be trustworthy (T17). That trustworthiness is derived in part from the trust placed on the underlying devices generating such data. IBN may use third-party sources, such as vendors that produce forwarding devices and developers who write software for network functions. An untrusted source can poison observed data, which can cause the optimization code to change the network in an undesirable fashion that is beneficial for an attacker.

Status Quo of IBN Security

Based on the aforementioned security challenges, we now discuss the current state of the art in IBN security. To the best of our knowledge, a comprehensive security assessment in IBN has not been realized. Toward that goal, we consider existing work related to IBN security and note security shortcomings.

Natural language approaches. LUMI[17] introduces a natural language-based IBN system, while presenting its open problems, such as ambiguities in conflicts and verifying correctness of natural-language intents. These uncertainties can pose unnoticed vulnerabilities in the IBN system, as several AI-based systems have brought about new types of security attacks (for example, adversarial data poisoning).

Network policy composition. Prior work has attempted to synthesize the diverse types of intents, such as reachability and waypoint,[29] bandwidth, stateful, and temporal policies,[3] as well as multi-tenancy.[31] These tools require formal specifications of intended behaviors, such as policy graphs or regular expressions. Since these tools focus on reconciling intents in predeployment, they do not consider any malicious intents from either an insider attacker or an attacker in a multi-tenant network.

Formal network verification. Network verification has long been studied within the networking community in the control plane[2,15,28] and the data plane.[6,13,16,19,21,25,36] Control-plane verification tools can verify diverse types of policies that described as functions or graphs.[2,28] However, these tools have not been implemented in the context of IBN security, which limits the classes of attacks they can formally verify.

Root cause analysis. Provenance has been proposed as a way to understand the causal dependencies of past intent and network activities.[32] Although such tools can observe and track relevant network state changes that can aid in understanding past attacks, they do not sufficiently solve the additional challenge of automatically identifying undesirable behavior (for example, side intents generation).

Vulnerability discovery. Dynamic testing tools[4,22] have been introduced toward discovering vulnerabilities and validating network configurations. While dynamic testing can reduce possibilities of novel attacks in IBN, it does not provide completeness guarantees (that is, absence of vulnerabilities).

Intent assurance. There have been several efforts toward automatically mining network specifications from the low-level network properties[9,20] and summarizing network behaviors in a natural language.[8] However, these efforts do not address malicious attackers in the data plane, which may generate malicious telemetry data to deceive the IBN controller. In addition, if the IBN system leverages AI models in the intent assurance, the system inherits the class of general AI-related security threats. For instance, security challenges arise in AI-driven zero-touch networks from poisoning attacks in the training phase to evasion attacks in the testing phase.[7]

Toward Secure Design Solutions and Research Directions

We discuss some possible directions to mitigate the aforementioned open security challenges. Most of the solutions either do not yet exist or are nascent ar-

eas of work, so we hope this discussion facilitates future research in the design of a secure and trustworthy IBN.

Securing intent fulfillment. *Security-privileged intents.* To allow for flexible network policies, IBNs should provide new intent types and properties that represent diverse, expressive, and flexible security policies. Since these security-related intents will impact the whole network, IBN can enforce multi-level intents that differentiate such security-related intents from regular intents installed by third parties or individual users. While users can create and observe their intents, a security module (that is, reference monitor) within the IBN controller should manage security-related intents akin to user-space and kernel-space privileges in traditional host-based operating systems.

Secure intent composition. During intent translation, IBNs can benefit from an additional *secure composition layer* to resolve intent composability. Although previous work has attempted to synthesize the diverse types of intents,[3,29,31] these tools often require complex inputs, such as policy graphs or regular expressions. Secure composition tools should be able to automatically recognize side intents and coarse intents.

Access control and information flow control. To tackle privilege-related challenges in IBN, new access-control and information flow-control mechanisms should be designed, similar to those used in various multi-user or multi-application environments (for example, host operating systems, mobile operating systems) but that take into account the unique security challenges posed by IBN.

IBN can benefit from a fine-grained, role-based access-control (RBAC) model to define permissions for each user. For example, one IBN implementation used in ONOS provides just three permissions (reading, writing, and event subscription) that provide an operator with nearly unrestricted control over all network intents. To further limit access, IBN should provide fine-grained controls that enable operators to only manage a subset of the overall intent space, similar to the concept of network slicing.

While ostensibly straightforward to implement, IBN introduces new challenges with RBAC because the mapping between intents and the underlying configuration (for example, flow rules, network protocols) must also follow the same access-control policy in spite of any optimizations (for example, multiple intents sharing the same underlying flow rules).

If the intent state is inconsistent with the intent processing, it will mislead a network operator or the intent controller, which can be used to attack control plane processes.

IBN can further construct intents to manage different intent spaces.[5] In such a model, each user can create intents within their virtual network. Those intents are then composed and deconflicted at the whole-network scale. Even if duplicate forwarding rules exist for intents in different intent spaces (for example, a forwarding rule that tunnels traffic between regions), the whole-network view can reconcile these flow rules globally.

RBAC alone, however, may be insufficient for properly controlling access in IBN. Previous work has shown that SDNs are vulnerable to cross-app poisoning attacks in which a malicious application can indirectly trick a privileged application to manipulate the control-plane state shared by both applications.[33] Since IBNs also run applications that create intents and subscribe to intent-related events, IBN is also vulnerable to similar attacks.

To avoid such vulnerabilities, IBN design should incorporate an internal security framework that functions as a reference monitor while enforcing information flow control (IFC) among intent-related control-plane decisions. Such a framework would monitor all requests from both users and applications to create intents, intent-related events subscribed to by the applications, and the subsequent operations.

In short, IBN should not only provide a complete permission model for both network operators and applications, but also protect against confused deputy attacks that could leverage data dependencies among different intent applications and the intent–forwarding rule mappings.

Securing intent assurance. *Intent verification.* Although network verification has been studied within the networking community,[2,28] the intent semantics and the network invariants need to be verified for IBN. Beyond verifying intent semantics among diverse intents, IBN may need to verify intents which cannot be defined by one of known properties, such as smart functions that evolve dur-

ing the intent lifecycle. Furthermore, IBN controllers can leverage incremental verification techniques to verify network invariants (that is, the absence of network degeneracy conditions) and pinpoint specific intents to blame if those invariants are violated. To avoid disclosing internal errors in processing intents, IBN must be able to distinguish the exposure of intent states depending on security levels. In addition, IBN should defend against the adversaries speculating the underlying network by probing with test intents.

Intent consistency. Ensuring consistency between the control-plane intents and data-plane forwarding rules is complicated by asymmetric operations (for example, rules that are installed some time after the creation of intents) and network dynamics (for example, rules that change according to underlying network events). Such a consistency should not be viewed as an invariant, but rather as a property that is *eventually* correct.

For such a property to be useful, however, consistency must be achieved prior to network changes; otherwise, the network will be perpetually in an inconsistent state. Periodic monitoring is thus a possible approach to ensure intent consistency (for example, the intent cleanup function in ONOS), but additional work is needed to make such monitoring code fast and scalable in a large network. *Backup intents* can also be leveraged as a fail-safe mechanism to resolve inconsistencies within an IBN.

Conclusion

Although programmable networking has made significant progress in the last few years, deployable IBN is still in its infancy as new features emerge and new standards develop. In this nascent phase, understanding the particular security challenges inherent to IBN affords the opportunity to realize a more secure and trustworthy IBN for the future.

In this article, we positioned IBN as a new, semantically rich paradigm for networking that can bring great opportunity into designing next-generation networks, yet unique security challenges must be addressed to enhance the assurance and trust placed in it. We studied why some of these challenges are hard to address with existing techniques and briefly outlined prescriptive solutions to tackle them. Addressing these security challenges will allow IBN to unleash its full potential in the coming years.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CNS-2339882. DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Texas A&M University under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Texas A&M University. C

References

1. 3GPP. Study on scenarios for Intent driven management services for mobile networks (Release 17), (2020); https://bit.ly/3K2ICqS.
2. Abhashkumar, A., Gember-Jacobson, A., and Akella, A. Tiramisu: Fast multilayer network verification. In *Proceedings of the 17th USENIX Symp. on Networked Systems Design and Implementation*, (2020), 201–219.
3. Abhashkumar, A. et al. Supporting diverse dynamic intent-based policies using Janus. In *Proceedings of the 13th Intern. Conf. on Emerging Networking Experiments and Technologies*, (2017), 296–309.
4. Alcock, P. et al. Improving intent correctness with automated testing. In *Proceedings of the 2022 IEEE 8th Intern. Conf. on Network Softwarization*. IEEE, 61–66.
5. Anjum, I. et al. Removing the reliance on perimeters for security using network views. In *Proceedings of the 27th ACM on Symp. on Access Control Models and Technologies*, (2022), 151–162.
6. Beckett, R. and Gupta, A. Katra: Realtime verification for multi-layer networks. In *Proceedings of the 19th USENIX Symp. on Networked Systems Design and Implementation*, (2022), 617–634.
7. Benzaid, C. and Taleb, T. AI-driven zero touch network and service management in 5G and beyond: Challenges and research directions. *IEEE Network 34*, 2 (2020), 186–194.
8. Birkner, R. et al. Net2Text: Query-guided summarization of network forwarding behaviors. In *Proceedings of the 15th USENIX Symp. on Networked Systems Design and Implementation*, (2018), 609–623.
9. Birkner, R., Drachsler-Cohen, D., Vanbever, L., and Vechev, M. Mining network specifications from network configurations. In *Proceedings of the 17th USENIX Symp. on Networked Systems Design and Implementation*, (2020), 969–984.
10. Cisco. *Cisco Intent-Based Networking (IBN)*; https://bit.ly/4aki6E0
11. Cisco. *Benefits of Intent-Based Networking*; https://bit.ly/4bjytlu
12. Clemm, A., Ciavaglia, L., Granville, L., and Tantsura, J. *Intent-Based Networking—Concepts and Definitions*, (2022); https://datatracker.ietf.org/doc/rfc9315/.
13. Dumitrescu, D., Stoenescu, R., Negreanu, L., and Raiciu, C. bf4: Towards bug-free P4 programs. In *Proceedings of the Annual Conf. of the ACM SIG on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 571–585.
14. Ferguson, A.D. et al. Orion: Google's software-defined networking control plane. In *Proceedings of the 18th USENIX Symp. on Networked Systems Design and Implementation*, (2021), 83–98.
15. Fogel, A. et al. A general approach to network configuration analysis. In *Proceedings of the 12th USENIX Symp. on Networked Systems Design and Implementation*, (2015), 469–483.
16. Horn, A., Kheradmand, A., and Prasad, M. *Delta-net: Real-Time Network Verification using Atoms*. In *Proceedings of the 14th USENIX Symp. on Networked Systems Design and Implementation*, (2017), 735–749.
17. Jacobs, A.S. et al. Hey, Lumi! Using natural language for intent-based network management. In *Proceedings of the USENIX Annual Technical Conf.*, (2021), 625–639.
18. Kang, M.S., Lee, S.B., and Gligor, V.D. The cross fire attack. In *Proceedings of the 2013 IEEE Symp. on Security and Privacy*. IEEE, 127–141.
19. Kazemian, P. et al. Real time network policy checking using headerspace analysis. In *10th USENIX Symp. on Networked Systems Design and Implementation*, (2013), 99–111.
20. Kheradmand, A. Automatic inference of high-level network intents by mining forwarding patterns. In *Proceedings of the Symp. on SDN Research*, (2020), 27–33.
21. Khurshid, A. et al. VeriFlow: Verifying network-wide invariants in real time. In *Proceedings of the 10th USENIX Symp. on Networked Systems Design and Implementation*, (2013), 15–27.
22. Kim, J., Ujcich, B.E., and Tian, D.J. INTENDER: Fuzzing intent-based networking with intent-state transition guidance. In *Proceedings of the 32nd USENIX Security Symp.*, (2023).
23. Kirkpatrick, K. Software-defined networking. *Commun. ACM 56*, 9 (2013), 16–19.
24. Liu, J. et al. P4v: Practical verification for programmable data planes. In *Proceedings of the 2018 Conf. of the ACM Special Interest Group on Data Communication*, 490–503.
25. Lopes, N.P. et al. Checking beliefs in dynamic networks. In *Proceedings of the 12th USENIX Symp. on Networked Systems Design and Implementation*, (2015), 499–512.
26. Juniper Networks. *Juniper Apstra*, (2020); https://juni.pr/3K37I9e
27. ONF *Intent NBI – Definition and Principles*; https://bit.ly/3WHvrD7.
28. Prabhu, S. et al. Plankton: Scalable network configuration verification through model checking. In *Proceedings of the 17th USENIX Symp. on Networked Systems Design and Implementation*, (2020), 953–967.
29. Prakash, C. et al. PGA: Using graphs to express and automatically reconcile network policies. *ACM SIGCOMM Computer Communication Rev. 45*, 4 (2015), 29–42.
30. Saltzer, J.H. and Schroeder, M.D. The protection of information in computer systems. In *Proceedings of the IEEE 63*, 9 (1975), 1278–1308.
31. Subramanian, K., D'Antoni, L., and Akella, A. Genesis: Synthesizing forwarding tables in multi-tenant networks. In *Proceedings of the 44th ACM SIGPLAN Symp. on Principles of Programming Languages* (2017), 572–585.
32. Ujcich, B.E., Bates, A., and Sanders, W.H. Provenance for intent-based networking. In *Proceedings of the 2020 6th IEEE Conf. on Network Softwarization (NetSoft)*, 195–199; 10.1109/NetSoft48620.2020.9165519
33. Ujcich, B.E. et al. Cross-app poisoning in software-defined networking. In *Proceedings of the 2018 ACM SIGSAC Conf. on Computer and Communications Security*, 648–663.
34. Ujcich, B.E. et al. Automated discovery of cross-plane event-based vulnerabilities in software-defined networking. In *Proceedings of the Network and Distributed System Security Symp.*, (2020).
35. Wood, L. Global intent-based networking market (2021–2026) by component, deployment, application, geography, competitive analysis and the impact of covid-19 with ansoff analysis. *Technical Report*. Research and Markets, (2021).
36. Zhang, P. et al. APKeep: Realtime verification for real networks. In *Proceedings of the 17th USENIX Symp. on Networked Systems Design and Implementation*, (2020), 241–255.

Jiwon Kim is a Ph.D. candidate in the Department of Computer Science at Purdue University, West Lafayette, IN, USA.

Hamed Okhravi is a senior staff member at MIT Lincoln Laboratory, Lexington, MA, USA.

Dave (Jing) Tian is an assistant professor in the Department of Computer Science at Purdue University, West Lafayette, IN, USA.

Benjamin E. Ujcich is an assistant professor in the Department of Computer Science at Georgetown University, Washington, D.C., USA.