Content-Based Security and Protected Core Networking with Software-Defined Networks

Konrad Wrona, Sander Oudkerk, Sebastian Szwaczyk, and Marek Amanowicz

The authors discuss the enforcement of content-based security policies at the different layers of the TCP/IP model, and introduce a proof-of-concept implementation of a content-based protection and release mechanism in a software-defined networking environment. Their aim is to provide consistent enforcement of security policies across multiple system layers and multiple security dimensions (confidentiality, integrity, and availability).

ABSTRACT

Successful execution of future network-centric military operations relies on effective enforcement of both need-to-know and responsibility-to-share principles. In modern military missions and coalitions, which have an increasingly agile character, a promising solution is to enforce security policies based on the properties of individual information objects — we call this approach content-based security. This article discusses the enforcement of content-based security policies at the different layers of the TCP/IP model, and introduces a proof-of-concept implementation of a content-based protection and release mechanism in a software-defined networking environment. Our aim is to provide consistent enforcement of security policies across multiple system layers and multiple security dimensions (confidentiality, integrity, and availability). The results of an analysis of a concrete example of a software-defined network emulated in Mininet are encouraging and confirm the effectiveness of our approach with respect to improving protection of data in transit. The work presented in this article offers a basis for further research in this area.

INTRODUCTION

The successful execution of future network-centric military operations relies on effective enforcement of both need-to-know and responsibility-to-share principles. The goal is to maintain fine-grained control over distribution of information, while ensuring that all releasable information is accessible to all authorized entities when required. Achieving this goal, considering the increasingly agile character of modern military missions and coalitions, requires the ability to enforce security policies at the level of individual information objects (and associated properties) — not only at the level of system-high IT systems or networks, as traditionally has been the case.

The ongoing evolution of modern IT systems in the direction of *software-defined everything* opens an opportunity for consistent enforcement of security policies across multiple layers of the system and with respect to the multiple dimensions of data protection, that is, confidentiality, integrity, and availability (C-I-A) [1]. Such multidimensional treatment of security is part of the protected core networking (PCN) [2] concept, as well as the focus of current NATO and NATO-nation activities related to data-centric security [3]. A particularly relevant technology in this context is software-defined networking (SDN) [4]. SDN technology also provides a suitable platform for implementation of PCN, both within individual colored clouds and in a federated environment, such as the one proposed for federated mission networking (FMN) [5].

Content-based protection and release (CPR) [6] is a NATO-specific approach to content-based security, developed to address some of the main challenges related to secure information sharing in FMN. In this article we show how the CPR approach can provide integrated protection of information in transit, and we discuss some implementation options for CPR enforcement in an SDN environment.

CONTENT-BASED SECURITY

In content-based security (CBS), security policies are dynamic and defined by the properties of the information (content) being protected. Although the intuitive application of CBS is access control and protection of confidentiality, CBS is also applicable to the integrity and availability dimensions. These dimensions are increasingly important because modern military operations are increasingly dependent on timely access to reliable information.

BACKGROUND ON ACCESS CONTROL MODELS

The traditional access control models for military operations, such as discretionary (DAC), mandatory (MAC), and role-based (RBAC) models, are not fully adequate in a modern federated mission environment, and do not offer the required expressiveness and granularity to enforce content-based security. The attribute-based access control (ABAC) model offers a powerful and unifying extension to these well-known traditional models. Under ABAC, requesters are permitted or denied access to a resource based on the properties, called attributes, that may be associated with users, resources, and context. Examples of attributes are identity, role, and military rank of users; identifier and sensitivity of resources; and, for context, time of day and threat level. Under ABAC, suitably defined attributes can represent security labels, clearances, and clas-

Konrad Wrona and Sander Oudkerk are with the NATO Communications and Information Agency; Sebastian Szwacz, and Marek Amanowicz are with the MIlitary University of Technology.

0163-6804/16/$25.00 © 2016 IEEE

sifications (for encoding MAC), identities and access control lists (for DAC), and roles (for RBAC). In this sense, ABAC supplements and encompasses traditional access control models [7]. Ensuring accuracy and trustworthiness of appropriate attributes and content labels is a critical requirement for enforcement of security policies, as discussed in [6].

Content-Based Protection and Release

CPR was developed by the NATO Communications and Information Agency specifically to address the challenges related to implementation of the FMN and future NATO operations. The model underlying CPR policies refines ABAC in two respects.

First, in addition to the attributes of users, resources, and context, the attributes of terminals are considered. Terminal attributes represent the capabilities of the device through which a user is trying to access a resource. Examples of terminal attributes are the hardware model, the type of encryption used to locally store data, and the type of protection used for the communication channel (e.g., IPsec or TLS).

Second, the CPR policies are structured in two distinct sub-policies called the *release policy* and *protection policy*. Both policies take into account resource and contextual attributes; the release policy also takes into account user attributes, whereas the protection policy takes into account terminal attributes. This enables the separation of policy management roles, and reflects the current procedures used within governmental and international organizations such as NATO. For example, consider the situation in which a user wants to access NATO classified information. In order to connect to the network infrastructure used for processing NATO classified information, a terminal must satisfy a number of technical requirements related to hardware and software configuration that are precisely defined in NATO technical directives and guidance documents. However, the security policy governing user access to the information stored in the network is defined in a separate set of directives and guidance documents.

More formally, a CPR policy states that a user u can access a resource r from a terminal t by checking if:

- The attributes of u and r satisfy the release policy P_R.
- The atributes of r and t satisfy the protection policy P_P.
- If the checks are both positive, *permit* is returned; otherwise, the result is *deny*.

Use Cases

Some use cases for CPR policies are presented in [6]. Here we briefly recount one of the use cases — a passive missile defense scenario — in order to give a concrete example of a CPR application. *Passive missile defense* focuses on mitigating the effects of potential missile attacks as well as those of anti-missile defense activities (e.g., terrain contamination due to missile interception). In the context of modern warfare, these effects often have a direct impact on large groups of civilians. Sharing this information in a timely and reliable way is thus of critical importance for effective civilian-military collaboration and protection of the population in conflict areas. An important aspect of such an information-sharing scenario is the multidimensional character of the security policy that must be enforced on the missile defense information originating from the NATO systems. Some parts of this information (e.g., the exact parameters of the missiles and launch locations) are not relevant to the humanitarian relief operation and should therefore not be released to external partners. On the other hand, information about the predicted fall-out area is of primary relevance and must be communicated to external partners in a timely and reliable manner. For this information, although confidentiality requirements are lower, the requirements for availability and integrity are high.

The C-I-A requirements described above are particularly applicable when a decision must be made about how the released data will be protected in transit between the point of origin (e.g., a database storing the missile defense information) and the client (e.g., an IT system belonging to mission partners such as a non-governmental organization). In particular, it would be advantageous to be able to choose a network route that provides an appropriate combination of C-I-A attributes. This type of dynamic risk-aware routing was explored earlier in the context of PCN [8], and is now becoming increasingly practical due to the widening deployment of SDN.

It would be advantageous to be able to choose a network route that provides an appropriate combination of C-I-A attributes. This type of dynamic risk-aware routing was explored earlier in the context of PCN [8], and is now becoming increasingly practical due to the widening deployment of SDN.

Cross-Layer Content-Based Protection and Release

Introduction

In the CPR architecture, management and enforcement of policies is based on a policy-based network management (PBNM) architecture with policy decision point (PDP), policy enforcement point (PEP), policy administration point (PAP), and policy information point (PIP) components. In this article it is assumed that all attributes and identity management information are available through the PIP and that a content-labeling infrastructure is in place. The core service in CPR is the CPR enforcement separation service (CPRESS) [6]. It is implemented by a PDP and one or more PEPs that act on the application layer of the TCP/IP model — the decision to grant access to an information object is made and enforced at the application layer. The CPR policy is administered by an application layer PAP.

CPR enforcement across the different layers of the TCP/IP model is referred to as "cross-layer CPR enforcement." Enforcing the CPR policy at another layer of the TCP/IP model means that the enforcement actions by the PEPs will be executed at that layer. In addition, there may be a need for a layer-specific PDP (i.e., a PDP that makes decisions for a group of PEPs acting on the same layer). In the case of both layer-specific PEPs and a layer-specific PDP, together they form a *layer-specific* CPRESS. (Note that a layer-specific PDP does not have to be implemented in the same layer as the PEPs with which it interacts.) A layer-specific PDP enforces a layer-specific policy. The translation of the general CPR

policy to layer-specific policies is illustrated in Fig. 1 and is assumed to occur at the application layer PAP. The layer-specific policies are then administered by layer-specific PAPs.

Layer-Specific Policy Enforcement

Once an information object is released to a user (and a terminal) at the application layer, the CPR policy can also be enforced during transport at lower layers of the TCP/IP model. The way in which the CPR policy is enforced by the layer-specific PEPs depends on attributes of the information object as well as protection mechanisms that are available in the particular layer of the TCP/IP model.

In the case of the network layer, a PEP can be integrated with the routing functionality, and the routing decision can be made based on route characteristics with respect to confidentiality (e.g., provision of an encrypted tunnel) or availability (e.g., the overall bandwidth or route reliability). The selection of a specific route is based on its characteristics (referred to as route attributes) and is determined by the CPR policy. The information that is used to determine the required route attributes can include content properties of the transported information (e.g., determining required C-I-A protection) but also user attributes (e.g., for traffic prioritization) and terminal attributes (e.g., for content aggregation). An example of policy enforcement at the network interface layer is a PEP integrated with the switching capability of a switch — the enforcement mechanism available to the PEP is the selection of the outgoing port (with associated link and link attributes).

Cross-layer enforcement of CPR policies introduces additional — and potentially substantial — overhead. In particular, it may not be efficient for a layer-specific PEP to, for each message the PEP mediates, request a decision from a remote layer-specific PDP. In such situations scalability can be improved by making use of a *local PDP* that is co-located with the PEP on a device. The local PDP can be viewed as an extension of the (remote) layer-specific PDP, which receives a copy of the layer-specific policy upon first creation, and policy updates afterward.

To further reduce complexity, the layer-specific policy can be translated to a device-specific policy, which does not contain requirements that are expressed for lower-layer attributes but instead identifies concrete device-specific policy enforcement actions that the PEP on the device has to take. This means the PEP no longer issues requests to the PDP, and policy evaluation is no longer required for each request. The device-specific policy can be generated either locally by a *local PAP*, or by the layer-specific PAP (after which it is distributed to the local PDP).

In our proof of concept (PoC) we use device-specific policies deployed at the SDN switches.

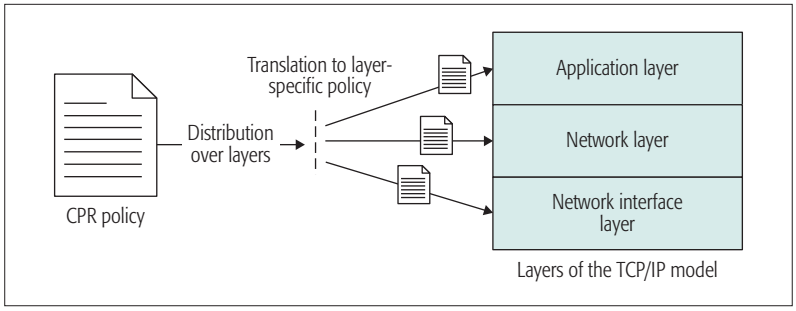

Figure 1. Translation of the CPR policy to the layer-specific policies.

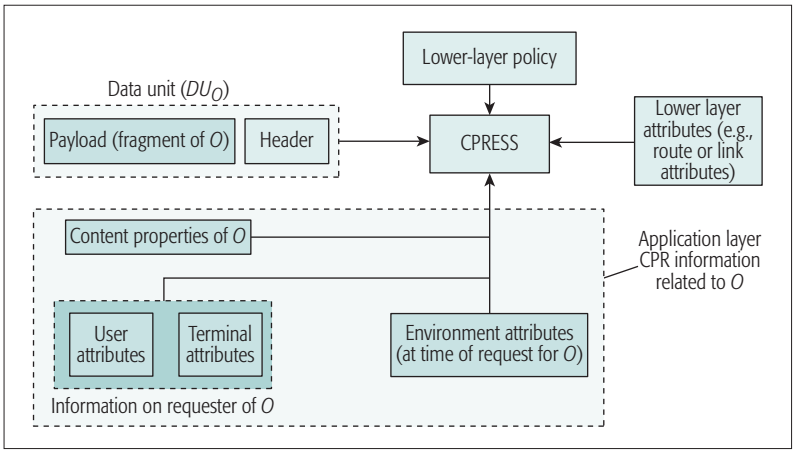

Figure 2. CPR enforcement at the network or network interface layer.

Lower-Layer CPR Policy Enforcement Based on Identifiers

At the network and network interface layers, the PEPs enforce a CPR policy on data units that carry a payload. If a request for an information object O at the application layer is granted, O will be transported by data units DU_O, each carrying a fragment of O. In order to enforce the CPR policy on DU_O, the (layer-specific) CPRESS requires access to the content properties of O as well as any other available attributes that are relevant to enforcement of the CPR policy for O. This combination of content properties and attributes is referred to as the application layer CPR information (ALCI) of O; see Fig. 2, in which the darker color of a box indicates a relationship with O. The applicable lower-layer attributes (e.g., route and link attributes) are assumed to be available to the layer-specific CPRESS.

Due to the fragmentation of O, the lower-layer PEP (or PDP) may not be able to determine the ALCI of O. In order to solve this problem in our PoC, we propose that the data units DU_O carry an identifier that represents the ALCI for O. A device-specific policy can then be expressed in terms of policy enforcement actions that are applied to DU_O carrying the identifier. In the context of a device-specific policy, we call this identifier an enforcement action identifier (EAI). The EAI is generated when the request for O is granted, and is administered by the application-layer PAP.

CPR in Software-Defined Networks

The objective of SDN based on CPR is to ensure that traffic follows a path through the SDN network based on the ALCI (or EAI) of the information that is transported. At the network interface layer this path is composed of links, and the lower-layer attributes are therefore link attributes.

The SDN architecture distinguishes three architectural components: an SDN application, an SDN controller, and an SDN switch. Figure

3 shows the correspondence between the SDN components and the components in the distributed CPR architecture: the SDN application and the SDN controller together implement the network layer interface PDP functionality, where the SDN controller deploys device-specific policies (i.e., flow table rules) on SDN switches. The SDN switches correspond to network interface layer PEPs that have a local PDP. Further, the translation of the network interface layer policy to the SDN switch policies takes place at the network interface layer PAP. This requires that the SDN controller request the network interface layer attributes (including information on associated ports or interfaces) from the SDN switch, after which they are forwarded to the PAP by the SDN application. The ALCI (or EAI) is provided by the application-layer PAP.

Applying CPR to Protected Core Networking

An example of a federated protected core network is depicted in Fig. 4. It consists of two protected core segments, PCS_1 and PCS_2. Each PCS has a PBNM system from which the CPRESS policies and components are managed. The information objects are hosted in a cloud environment that is reachable through PCS_1. The user u and terminal t requesting an information object O are part of a colored cloud, CC, that is connected to PCS_2. PCS_1 and PCS_2 are networks comprised of links and each link has *link properties*. The term *link properties* is used in the PCN model and corresponds to the CPR term *link attributes*. Once O is released to u (and t), the data units DU_O follow a path P composed of links through PCS_1 and PCS_2. For all links in P the link attributes meet the link requirements as determined by the CPR policy. The link requirements can be explicitly included in the CPR policy or defined in the form of bridge predicates [9]. Note that in Fig. 4 the lower-layer CPRESS is active in the PCS, whereas the application layer CPRESS is active in the cloud environment. In the PCS it is assumed that the PBNM architecture contains a central lower-layer PDP (and PAP) and that the lower-layer CPRESS within the PCS is composed of PEPs and PDPs local to the lower-layer devices (e.g., router, switches). In Fig. 4 the lower-layer CPRESS is then managed from the PBNM system, and there is a logical interface *CPR_INT* between the application layer CPRESS and management system $PBNM_1$. In Fig. 4 it is assumed that the cloud environment and PCS_1 are managed by the same authority. Further, it is assumed that there is a management interface, *M_INT*, between $PBNM_1$ and $PBNM_2$ (not shown in Fig. 4).

The example PCN architecture from Fig. 4 can be implemented using SDN technology. In that case the SDN controllers of PCS_1 and PCS_2 exchange the link requirements (which determine the path p) over *M_INT*. In practice this means that if the controller in PCS_2 receives a data unit (packet) DU_O with an EAI that it cannot map to link requirements, it will request the requirements from the controller in PCS_1, after which it can program flow table rules on the SDN switches in PCS_2 (and DU_O will follow the path p).

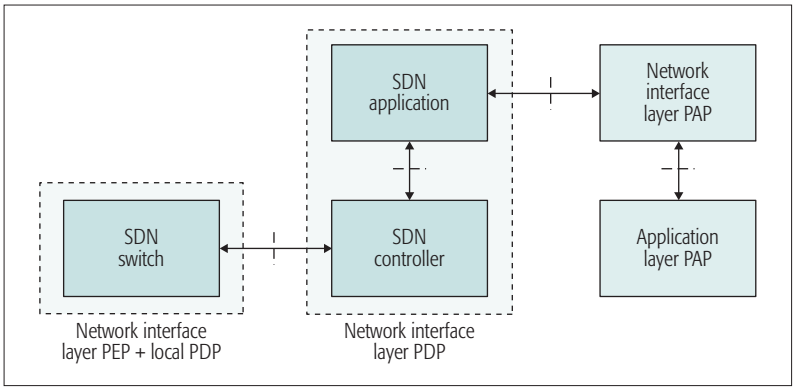

Figure 3. SDN components in the CPR architecture and their interfaces.

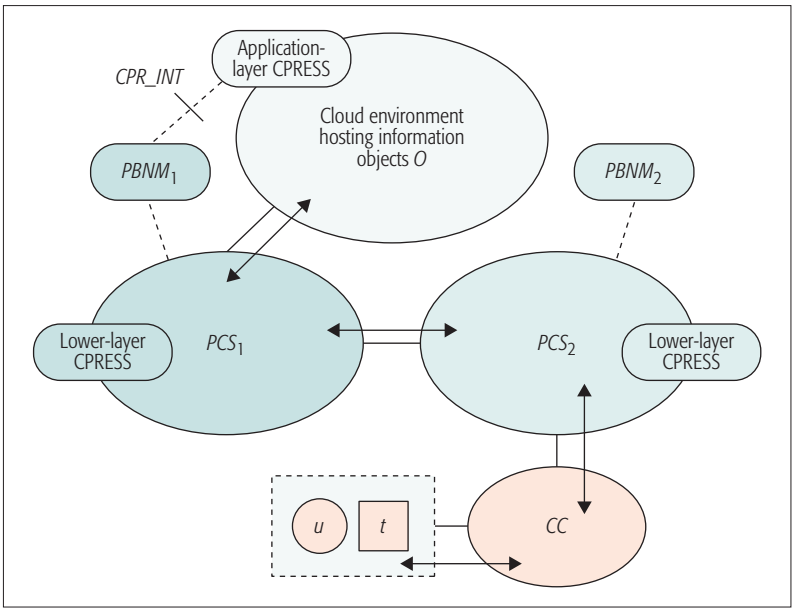

Figure 4. PCN architecture in which DU_O follows a path through two protected core segments PCS_1 and PCS_2 in order to reach user u and terminal t.

Proof of Concept

Implementation

In our PoC — the implementation of a CPRESS in an SDN environment — policy enforcement is based on the use of an enforcement action identifier, EAI_O, which determines the path that DU_O can take through the SDN network. As there may be more than one path that meets the requirements of the CPR policy for O, the identifier EAI_O essentially determines a class of paths. Therefore in our PoC it is referred to as a *path class identifier*.

The integration of CPR and SDN in our PoC is simplified as follows:

- The network interface layer PDP consists of the SDN controller only.
- The functionality of the SDN application is integrated with the application-layer PDP.
- The network interface layer PAP and application-layer PAP are merged into a single PAP.
- The PAP and application-layer PDP are merged into one PDP component.

Furthermore, the device-specific policy (i.e., the policy that constitutes the flow on the SDN

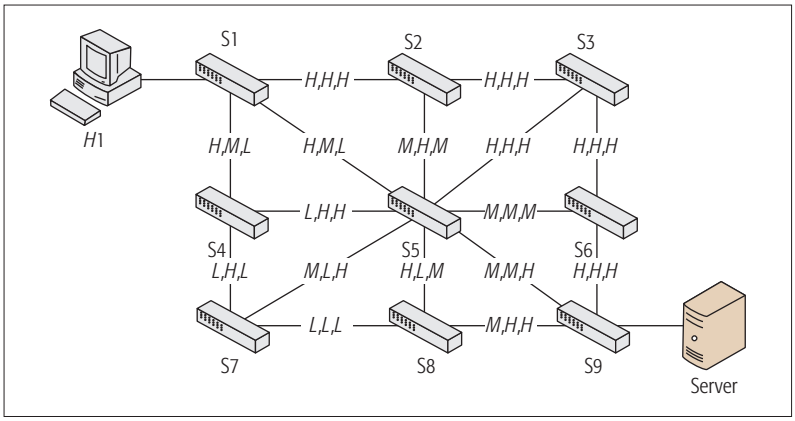

Figure 5. The test network topology.

switches) is not generated at the PAP but by the SDN controller itself. In order to support the use cases, the implementation includes an application-layer PEP that mediates access requests for information objects. This PEP and the PDP together form the application-layer CPRESS, which, due to the inclusion of the SDN application in the PDP, is referred to as the SDN CPRESS.

The set of paths identified by the path class identifier is configured in the network based on the link attributes. Examples of link attributes are the available bandwidth, the physical medium, and the available network protection measures. Our innovative approach is to derive the requirements for the link attributes dynamically by applying the CPR policy based on the content properties of the transported data.

The implementation of the PoC is independent of the types of link attributes. The PoC uses the three category link attributes *Confidentiality*, *Integrity*, and *Availability*. Here, "category" means that each attribute represents a set of representative link attributes (given the type of SDN devices used) which establish the objective of the category (i.e. one of the C-I-A). Selection of the representative sets of link attributes, given the type of SDN devices used, is a topic for future research. The category link attributes can take the values *None*, *Basic*, *Standard*, *Enhanced*, and *High*. In the case of availability the value *None* is interpreted to mean that the link is down, that is, it cannot be used and is therefore not considered for transport of information.

Transferring the Information via Protected Paths

The information that is transferred through the SDN network is contained in a packet stream in which packets correspond to data units DU_O as discussed above. The first packet in the packet stream contains the path class identifier, and when the switch receives this packet, it looks up the flow table to find the output port for the packet. If a matched flow entry does not exist, the switch sends the packet to the controller. The controller parses the packet and looks for the path class identifier. If the controller knows the path class identifier, it can produce a set of possible paths through the network based on its knowledge of the network topology. Otherwise, the controller communicates with the SDN CPRESS to obtain the link requirements identified by the path class identifier. If more than one possible path exist (i.e., it is composed of links of which the link attributes meet the link requirements), the controller must choose one of them but must store all possible paths in order to be able to react quickly if a chosen path fails. Furthermore, the controller programs the flows on switches using the path class identifier, source and destination IP addresses, and source and destination TCP ports. These five attributes uniquely identify every connection between the requester and the server. After the flows are programmed, the controller returns the packet to the network, and the switches forward the packet to the requester based on the flows programmed. When a switch receives the second packet in a packet stream, it forwards this packet based on the matched flow entry, and so on for all subsequent packets.

Example of a Network Instance

An application of our approach to a concrete network instance of the PoC emulated in Mininet and depicted in Fig. 5 is described below. The emulation environment is composed of a Floodlight controller and OpenVSwitch 2.1.1 switches, and can be viewed as representing a protected core network consisting of one PCS. Each link is characterized by the three category link attributes described earlier with values explicitly signifying the level of protection provided with regard to each of the C-I-A dimensions. The set of values is simplified however and is comprised of *high* (*H*), *medium* (*M*), and *low* (*L*). The OpenFlow specifications (available at the time of writing) specify only a few link attributes. In order to allow for the use of the C-I-A link attributes, we propose extending the values transported in the OpenFlow PORT_STATUS_REPLY message. This can be done without further modification to the protocol.[1] In our concrete example we assume that a host *H*1 generates requests for information to a *server* (Fig. 5). To this end a specific host program was implemented that emulates requests; the implemented server program also emulates the behavior of the application-layer PAP, PDP, and PEP.

Without the use of the path class identifier, the path between the host and the server is configured using the standard Floodlight shortest path algorithm. Clearly, the links that form the shortest path will not always meet the protection requirements for the transported information; however, the chosen path will always involve the minimum number of links.

In contrast to when the path class identifier is used to configure the network path, the resulting path will always meet the protection requirements; however, it may be substantially longer than the shortest path.

In order to illustrate this trade-off, we performed a simple analysis for the test network from Fig. 5. We assumed that the server stores data with largely varied content properties and thus covers all possible transport protection requirements, that is, all 27 possible combinations of values {*L*, *M*, *H*} in the C-I-A space. The plausibility of some such combinations (e.g., whether it makes sense to have data requiring high confidentiality and availability but low integ-

[1] For a more detailed discussion of implementation challenges, see [10].

rity) is disregarded here. We further assumed that host $H1$ requests all possible types of data stored at the server, resulting in the need to set up network paths meeting every possible combination of protection requirements. If the standard shortest path approach is used, some packets will be transported over links not meeting their protection requirements; the proportion of these incorrectly protected packets for each of the C-I-A dimensions in our simple network is listed in Table 1. These values can be interpreted as rough indicators of the security risk involved in transmission of the data over the network. For the particular network configuration depicted in Fig. 5, only 15 percent of possible data protection requirements would be fulfilled if a naive shortest path algorithm is used. This means that if a uniform distribution (of transported information) among all possible protection requirements is assumed, a probability that at least one of the protection requirements would not be met for a particular information object is 85 percent. The use of the path class identifier mitigates this risk.

Furthermore, we performed a similar evaluation assuming all possible configurations of attributes for the shortest path (i.e., $(3^3)^2 = 729$ possibilities). The quantitative results, reported in Table 1, were similar to the ones obtained for a single network configuration.

Table 1 also illustrates the average additional use of links when using the path class identifier compared to the shortest path approach. The average additional overhead (or cost of protecting information using CPR) introduced by using the path class identifier for all possible types of content for the specific static network depicted in Fig. 5 is 69 percent. We also performed an evaluation for a dynamic case, in which the C-I-A attributes of the links were progressively degraded from their depicted initial values up to the moment when still at least one CPR-compliant path was available. Such a scenario could be interpreted as an evolution of a non-maintained network over time, due to increasing attacks or new vulnerabilities being discovered. The average CPR-induced path extension was in this case 67 percent.

This cost appears to be acceptable given the significant reduction of risk of compromising the transported information.

Number of paths not meeting protection requirements (shortest path algorithm)				
Network	Confidentiality	Integrity	Availability	Total
Static	9 (33%)	9 (33%)	18 (66%)	23 (85%)
Dynamic	9477 (48%)	9477 (48%)	9477 (48%)	16939 (86%)

Average path length (in number of used links)		
Network	Shortest path	Path class identifier
Static	2 (100%)	3.37 (169%)
Dynamic	2 (100%)	3.34 (167%)

Table 1. Analysis results for Fig 5.

Related and Future Work

A recent survey of research topics related to the security of SDNs can be found in [11, 12]. Most of the earlier work focused on access control and enforcement of a security policy with respect to network flows as opposed to individual information objects. Also, to our knowledge, the earlier approaches did not target a complete cross-layer integration of SDN security mechanisms with application-layer security, which is one of the main objectives of the content-based security approach. For example, the Ethane [13] and Resonance [14] architectures include the enforcement of policies based on pre-registered hosts, protocols, switches, users, and access point names. However, they do not take into account the content properties, which is critical for enforcement of CPR policies. Some of the earlier work on SDN security addresses security of the SDN components themselves [15] or offers monitoring functionality [16], both of which can accompany preventive security measures offered by our CPR approach.

We have presented a high-level description of an implemented PoC that integrates CPR enforcement in an SDN environment and can be applied to PCN-compliant systems. Our aim is to address the need for consistent enforcement of security policies across multiple layers of the TCP/IP model with respect to all C-I-A dimensions of information protection. This need has been identified as an important requirement for future military network-centric operations. We have presented a concrete instance of a CPR-enabled SDN network, which was emulated in Mininet, and demonstrated how the effectiveness of our approach can be assessed. The initial results are encouraging; however, a substantial amount of additional work is required to assess the performance implications of our approach. This would involve both emulation and simulation of a larger set of realistic networks and the investigation of real SDN implementations. Our future plans include validation of our solution in a larger-scale implementation in order to investigate performance (e.g., in terms of throughput and latency) as well as security management overhead. We would also like to investigate possibilities for integrating our approach with software defined exchanges (SDX) [17] in support of federated PCN and FMN deployments. Finally, attention must be paid to ensuring the proper assurance level and secure configuration of components involved in cross-layer CPR enforcement.

References

[1] K. Wrona and S. Oudkerk, "Integrated Content-Based Information Security for Future Military Systems," *Proc. IEEE MILCOM*, 2015, pp. 1230–35.
[2] G. Hallingstad and S. Oudkerk, "Protected Core Networking: An Architectural Approach to Secure and Flexible Communications," *IEEE Commun. Mag.*, vol. 46, no. 11, 2008, pp. 35–41.
[3] J. Melrose *et al.*, "Labelling for Integrity and Availability," *Proc. Int'l. Conf. Military Commun. and Info. Systems*, 2016.
[4] D. Kreutz *et al.*, "Software- Defined Networking: A Comprehensive Survey," *Proc. IEEE*, vol. 103, no. 1, 2015, pp. 14–76.
[5] A. Domingo and H. Wietgrefe, "On the Federation of Information in Coalition Operations," *Proc. IEEE MILCOM*, 2013, pp. 1–8.
[6] K. Wrona and S. Oudkerk, "Content-Based Protection and Release Architecture for Future NATO Networks," *Proc. IEEE MILCOM*, 2013, pp. 206–13.
[7] X. Jin, R. Krishnan, and R. Sandhu, "A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC," *Data and Applications Security and Privacy XXVI*, Springer, July 2012, pp. 41–55.

[8] K. Wrona and G. Hallingstad, "Real-Time Automated Risk Assessment in Protected Core Networking," *Telecommun. Systems*, vol. 45, no. 2–3, 2010, pp. 205–14.
[9] A. Armando *et al.*, "Compiling NATO Authorization Policies for Enforcement in the Cloud and SDNs," *Proc. IEEE Conf. Commun. and Network Security*, 2015, pp. 741–42.
[10] S. Szwaczyk, K. Wrona, and S. Oudkerk, "Implementation of Content-Based Protection and Release in Software Defined Networks," *Proc. Nat'l. Symp. Telecommun. and Teleinformatics*, Cracow, Poland, 2015, pp. 1099–108.
[11] I. Ahmad *et al.*, "Security in Software Defined Networks: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 4, 2015, pp. 2317–46.
[12] I. Alsmadi and D. Xu, "Security of Software Defined Networks: A Survey," *Computers & Security*, vol. 53, 2015, pp. 79–108.
[13] M. Casado *et al.*, "Rethinking Enterprise Network Control," *IEEE/ACM Trans. Net.*, vol. 17, no. 4, 2009, pp. 1270–83.
[14] A. Nayak and A. Reimers, "Resonance: Dynamic Access Control for Enterprise Networks," *Proc. Wksp. Research on Enterprise Networking*, 2009, pp. 11–18.
[15] P. Porras *et al.*, "Securing the Software-Defined Network Control Layer," *Proc. Network and Distrib. System Security Conf.*, 2015, pp. 1–15.
[16] J. R. Ballard, I. Rae, and A. Akella, "Extensible and Scalable Network Monitoring Using OpenSAFE," *Proc. 2010 Internet Network Management Conf. Research on Enterprise Networking*, 2010, pp. 1–6.
[17] A. Gupta *et al.*, "SDX: A Software Defined Internet Exchange," *Proc. ACM Conf. SIGCOMM*, 2014, pp. 551–62.

Biographies

KONRAD WRONA [SM] (kwrona@aol.com) is a principal scientist at the NATO Communications and Information Agency in The Hague, The Netherlands. He has extensive work experience in industrial and academic research and development environments. He received an M.Eng. in telecommunications from Warsaw University of Technology, Poland, in 1998, and a Ph.D. in electrical engineering from RWTH Aachen University, Germany, in 2005. He is an author and a co-author of over 60 publications, as well as a co-inventor of several patents. His professional interests focus on secure design of a broad range of communication and computing systems. He is a Senior Member of the ACM and a member of the IACR.

SANDER OUDKERK (sander.oudkerk@agentsierra.nl) is the founder of Agent Sierra Consultancy Services, an IT security company based in Amsterdam, the Netherlands. He received an MSc. degree in mathematics from Utrecht University, the Netherlands, in 2000. He has over 15 years' experience in information assurance for military and defense applications. He specializes in information leakage prevention solutions and cyber security concept development.

SEBASTIAN SZWACZYK (sebastian.szwaczyk@wat.edu.pl) received his M.Sc. degree from the Military University of Technology, Warsaw, Poland in 2015 in telecommunication engineering. Currently he is a PhD student in the Military University of Technology, Warsaw. His research interests include software engineering, computer engineering, communications protocols, network management, and virtualization.

MAREK AMANOWIC (marek.amanowicz@wat.edu.pl) is a professor at the Military University of Technology, Warsaw, Poland. He received his M.Sc., Ph.D., and D.Sc. degrees from the Military University of Technology in 1970, 1978, and 1990, respectively, all in telecommunication engineering. He has been engaged in many research projects, especially in the fields of communications and information systems engineering, mobile communications, communications and information systems modeling and simulation, communications and information systems interoperability, network management, and electronic warfare. He is currently working in the area of military communications and networks with focus on information assurance and cyber defence.