An ABAC-based Policy Framework for Dynamic Firewalling

Sören Berger*, Alexander Vensmer† and Sebastian Kiesel‡
*Computing Center
University of Stuttgart
Stuttgart, Germany
e-mail: soeren.berger@rus.uni-stuttgart.de
†Institute of Communication Networks
University of Stuttgart
Stuttgart, Germany
e-mail: alexander.vensmer@ikr.uni-stuttgart.de
‡Computing Center
University of Stuttgart
Stuttgart, Germany
e-mail: sebastian.kiesel@rus.uni-stuttgart.de

***Abstract*—This paper presents the Policy Framework of DynFire, a novel approach for attribute-based, dynamic control of network firewalls. DynFire allows an individually controlled, secure access to IT resources of a large organization, with particular focus on mobile users and users with restricted rights, such as subcontractors. The basic assumption behind DynFire is that, within a secured network domain separated from the Internet, a temporary binding between an IP address and a single user ID can be established. Users with different attributes can authenticate to the network and get individual access to network resources. To administrate such a large amount of users and different access rights within a secured network domain of an organization, which includes distributed organisational zones, a policy framework is needed. The following paper presents a policy framework for dynamic and distributed firewalls which is able to grant access control on a per-user basis, with multitenancy capabilities and administrative delegation.**

Keywords-dynamic firewall control; network security; policy based network access control.

I. Introduction

Firewalls are a well-understood and widely deployed means of protecting IP networks [1]. Their use is based on the assumption that the network can be divided into distinct domains with different security requirements and threat levels [2]. Located at domain boundaries, firewalls forward or reject network traffic between these domains, according to security policies that are usually configured statically into the firewalls [3]. However, this assumption, and thus the applicability of firewalls, is increasingly challenged. With the widespread use of mobile wireless as well as remote access over the Internet, domain borders get more and more blurred.

A mobile user changing from one access network to another, usually, receives a new IP address randomly chosen from an address pool. Therefore, IP packets, *usually*, do not carry enough information for a firewall to perform user-based access control decisions. While this puts the usefulness of firewalls into question, other developments reinforce the need for them. As the operator of a large campus network, we encounter an increasing number of devices in our network, which are not “classic” telecommunications or office PC equipment. This includes, e. g., building automation systems or scientific measurement devices. While these systems are often vulnerable due to missing or outdated security mechanisms (e. g., operating system updates, virus scanners, password policies, etc.), they also have an increased need for remote access, e. g., for maintenance technicians. Placing a firewall in front of such systems may improve security, but conventional firewalls with static policies are not flexible enough for fine-grained access control. Wide spread firewalls and distributed administration added new requirements to enterprise institutions firewall scenarios. This paper presents the Policy Framework of DynFire, a new framework for the

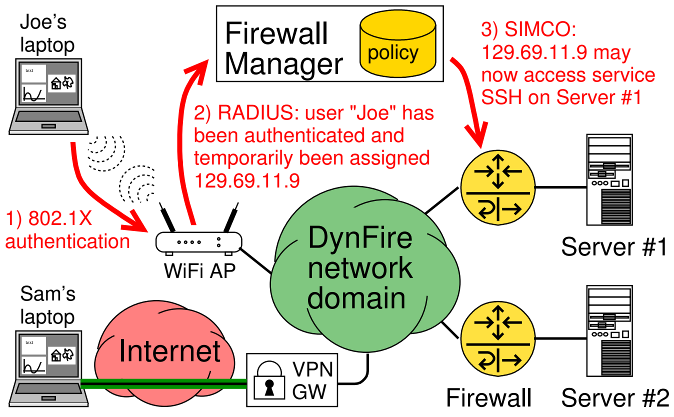
Joe's laptop
Firewall Manager
policy
3) SIMCO: 129.69.11.9 may now access service SSH on Server #1
2) RADIUS: user "Joe" has been authenticated and temporarily been assigned 129.69.11.9
1) 802.1X authentication
WiFi AP
DynFire network domain
Server #1
Sam's laptop
Internet
VPN GW
Firewall
Server #2

Figure 1. DynFire scenario

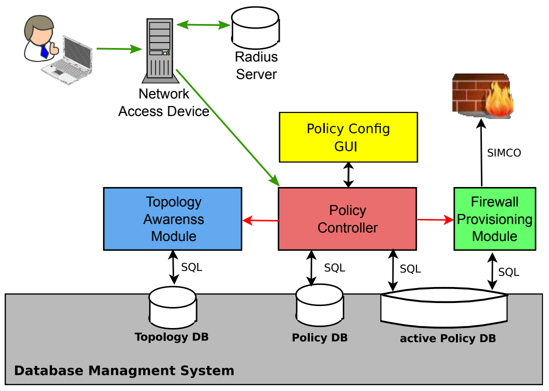

Figure 2. DynFire architecture

dynamic attribute-based configuration of firewalls. The rest of the paper is structured as follows. Section II describes the system architecture of DynFire. Section III present the Policy Framework. Section IV summarizes the related work. Section V concludes the paper and summarizes further steps.

II. DynFire System Architecture

The goal is to create a policy framework for the architecture DynFire to provide dynamic firewalling. It can perform attribute-based access control to decide which user can access firewall secured resources. DynFire as such cannot be a solution for the whole Internet. Instead, it can be used to secure the IT resources of a single organization. It is assumed that this organization operates a network that is protected from the Internet by firewalls. The DynFire administrators do not create firewall rules based on individual IP addresses, but describe the desired communication relationship between users and services (network resources) in an administration panel in the form of policies.

Figure 1 shows the concept of DynFire. When a user logs into the network he receives a temporary IP address. The “Firewall Manager” uses the temporary binding between user and IP address and the policies to create firewall rules, which are related to a single device used by a single user. Those firewall rules are valid as long as this particular user is logged in. In a complex network topology the Firewall Manager has to configure all firewalls on the paths between the users and the resources, respectively. The architecture is shown Figure 2 and each main component is introduced in this section.

A. Policy Config GUI

The Policy Config GUI is the central instance to administrate all policies of the system. Every administrator and owner is able to login and control its assigned zones. Additional administrators have the possibility to enter policies for their users in very comfortable way as already described.

B. Network Access Device

The network access device is responsible for authenticating and logout of the user. It creates context information about the user e.g., from which location the user is logged in. This context information will be carried along the username and the IP address to the policy controller.

C. Policy Controller

When the Firewall Manager is notified about the login and temporary binding between UserID and IP address is received. It executes an access evaluation algorithm with will be described later in this paper. The algorithm returns a set of permissions, which permits the user to access all for him enabled services. These permissions contain the IP address of the resource, the port of the service, and the transport layer protocol (TCP or UDP) used by the service. The source IP address was passed by the network access device. Firewall 5-tupel rules can be generated from these data. In specific cases where protocols using dynamic ports are involved, additional information about the service is sent together with it. These additional information advices the firewalls to deploy this firewall rule in a special manner. For example, the firewall has to activate a connection tracking module to be aware of related connections. These related connections have to be accepted as well. Information about the concerned firewalls is retrieved from the Topology Awareness Module. Then, the rules are sent to the Firewall Provisioning Module. In case of a logout the Policy Controller remove all associated rules for the user.

D. Topology Awareness Module

The Topology Awareness Module has to find all firewalls on the path between two given hosts. Therefore, it has to know the network topology. The current version is able to work with a static topology map. An advanced version that can detect the topology automatically based on LLDP (Link Layer Discovery Protocol) [4] and SNMP (Simple Network Management Protocol) [5], is currently under development. It will also interact with the routing protocol, in order to configure firewalls on the alternative path, in case a rerouting occurs.

E. Firewall Provisioning Module

The Firewall Provisioning Module is responsible for transferring firewall rules to a set of firewalls. Several protocols for firewall control exist [6]. The SIMCO protocol [7] was chosen, because of its flexibility and simplicity. Several SIMCO implementations for Linux (iptables), Cisco, and Juniper routers are currently under development or testing. Furthermore it is possible to integrate the Firewall Manager into the Astaro Command Center [8], which provides an integrated firewall solution. This multitude of supported firewalls allows DynFire to be deployed in heterogeneous network environments.

III. Policy Framework

Common access models like RBAC [9] or ABAC [10] (which is used in this work) are very abstract specified models and not suitable for an implementation. They do not deal with the problem "HOW" a rule can be evaluated. If one does not find a suitable representation of your rules, the model still works but it will become nearly impossible to administrate, because rules could become to complex to define by network administrators. However, ABAC is a generic model that describes neither how the attributes and the evaluation function look like, nor how the evaluation process works. We developed a representation of high-level policies, which provide an intuitive way to grant access to firewall secured resources and an efficient way to evaluate firewall rules.

A. Capabilities

In the following, each main feature is described in detail and why it is needed.

1) Policy Definitions: In DynFire, a policy describes how a user can access a resource or a server. In particular:

> Under which **condition** a **user** can access a **service** on a **resource**.

A typical example would be a policy like:

> "Bob can access the wiki of his institution during daytime."

DynFire adds additional possibilities for a fine granular definition of conditions. So, it could also be possible to define a policy like:

> "Alice can access the ssh-server only if the intrusion detection system of the network set its client health value 5 or more and she is connected via VPN, but not via WiFi."

The Policy Framework refers to three different terms here. The **Who** is already determined during the login process. The **condition** is a logic that grants access if all associated credentials match.

Services: DynFire associates services with subsets of a host. In common firewall understanding, a service is described by its port. Typically, this is also true for DynFire; but, in addition, there can be more information to describe abstract services like âIP-telephonyâ.

Not all firewalls may support all different type of services. So the interpretation of the service description has to be done at the provisioning level of the DynFire firewalls.

Resources: A resource is typically a host or a group of hosts where a service is running on. The hosts themselves are identified by their IP addresses. This means that resources are typically a set or range of IP addresses.

2) Multitenancy Capabilities and Delegation: As described in the introduction of this paper, DynFire is targeted at medium to large scale networks. In particular, this means the administrative tasks like granting access to a resource can not be done by a single administrator, especially if the corresponding Resources are not within his administrative domain. Following the structure of big organizations, DynFire divides the network into different parts based on their networks and subnets.

This means that the top administrator owns all network resources and may split them into different parts. Each part is represented as a subset of the original network. Those subsets can be assigned to other administrators that can assign policies for their subset of the network but not for other subsets. This allows DynFire to provide a scalable administrative hierarchy that decentralizes the responsibilities in the network.

3) User Groups: To handle a potential big number of users, DynFire administrators can group users. Besides normal assignments of access control to users, the same should be possible for groups. Groups may defined by any administrator in their administrative domain. It is also possible to add groups to other groups and create recursive group definitions for better usability and scalability of the group definitions.

4) Context-sensitive Rule Evaluation: Since mobility has has grown importance, this is reflected by adding context-sensitive functionality to the framework. A user may access the network via different networks and devices. This can be reflected in the defined rules, like allowing unencrypted mail access only over a secured connection.

5) Audibility: In distributed administrated network it should always be possible to track the changes and know who is responsible for the changes. In DynFire, it is possible to restore any state of the Policy Framework in the past a check who changed the configuration. This is mostly motivated by legal requirements and in order to allow analysis during or after attacks on resources in the network.

B. Components

The Policy Framework implements some basic components required for the policy specification and rule evaluation. It is capable to extend this list by any parameters users provide during the login process.

DynFire uses some terminology that will be described in this section. Most of the terms are much overloaded in the literature. We tried to align the common understanding of those terms with the functionality of DynFire. Figure 3 shows a possible implementation of the frameworks. The different entities will be described in this section.

1) Credential: ABAC is based on attributes. In this case, the attributes are represented by credentials. Credentials can be assigned to users by administrators or they are formed during the login process of users. Also, every permission,

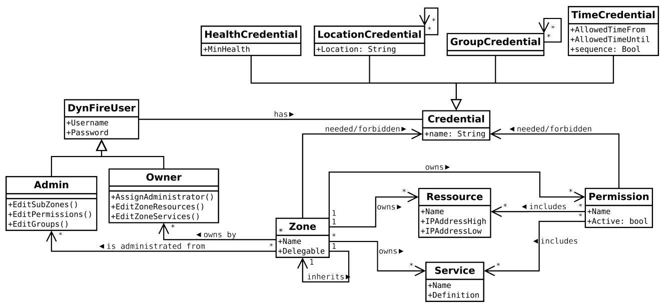

Figure 3. DynFire Policy Model

which is required for getting access on a service running on a corresponding resource, has a list of credentials users need to have for getting access. In addition, there is a list including forbidden credentials. Thus, if a user possesses a credential which is element of=20 such a list, it is not possible for one to get such a permission. Each credential technically is evaluated in the same way. This means that there can be additional credentials added over time, especially without adding additional source code in the policy evaluator. The design itself allows adding additional parameters that must (or must not) match the user credentials.

The implementation of DynFire currently considers four kinds of Credentials:

(a) Group credentials
A group only allows access to a resource if the user is in the required group. As shown in Figure 3, a group can contain itself. This results in the possibility to implement recursive groups. The side-effects, like looped groups, are resolved in the database through stored procedures which is the common solution to keep data consistency.
(b) Location credentials
A location credible typical is set in the login process. This information is typically passed from the login program. For example, if the user logs in via VPN, the VPN login process can set the parameter. Based on this information the user may be granted access to the resources or not. Other possible location credentials are 802.1X [11] (e.g., WiFi) or any other network authentication method to map an IP address to a user.
(c) Health credentials
An external entity provides information about the host that the user uses to login. During the policy evaluation process, this value is checked against the specification in the policy and have to match the minimum value. For example, a intrusion detection system can scan the host for possible security holes or the Network MAC address is known by the anti-virus software of the network and provide the information that the logged in system runs the newest virus scanner.
It can be a simple value like “secure” and “not secure”. It can also provided a fine granular specification like a range from 1-100, 10 for example, would means a low security status.
(d) Time credentials
A time credential enables the specific permission only during a fixed time. Typically, this can be used to allow access to a resource only during the working days.

2) Permission: A permission in the Policy Framework is a combination of:

(a) required credentials
(b) forbidden credentials
(c) resources
(d) services

This means that the users, which possess all needed credentials (a), and none of the forbidden ones (b), have access to the resource (c) via the service (d). A simple example is that the user Bob from VPN can access the server 192.168.123.15 via ssh if he is belonging to group Administrators and did not login via WiFi.

3) Zones: To separate different subnets from each other the term “zone” is introduced. A zone is collection of network or IP ranges. The root zone contains the complete network managed by the Policy Framework. This network can have some child that contains a subset of the network(s) of this root zone. Let

$$M_f = \{net_{f1}, net_{f2}, net_{f3}, ...\} \tag{1}$$

be zone where net_x is a network in CIDR notation with the relation $\subseteq$. A network $n_i \subseteq n_j$, when all IP addresses of the network n_i are in the network n_j. Let

$$M_c := \{net_{c1}, net_{c2}, net_{c3}, ...\} \tag{2}$$

be child zone of M_f. Then following is true:

$$\forall net_{cx} \in M_c, \exists net_{fn} \in M_f : net_{cx} \subseteq net_{fn} \tag{3}$$

This means that each network in a sub-zone has to be in one of the networks in the parent-zone. This also implies that a network may be member of multiple sub-zones. Figure 4 shows a simple example of a zone structure. Each zone has its own required and forbidden credentials to define what credentials are needed to access the resources of this zone.

Admins and Owners: Administrators and owners are two central administrative entities in the Policy Framework. Each zone has a least one administrator and one owner. Both have different roles in the framework. An owner is - as it says - the owner of the sub-zone. This typically, does not mean that this user administrates the zone. He has the ability to assign administrators to his zone and split the zone in different sub-zone for further delegation.

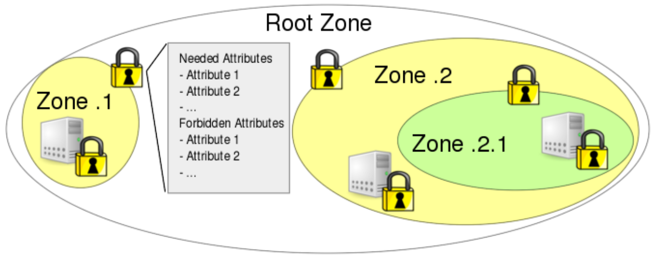

Figure 4. DynFire zone structure example

An administrator defines local groups, services, resources and finally permissions to a zone. Of course, an owner of a zone can assign the administration tasks to his own account.

C. Access Evaluation Algorithm

In contrast to other access control systems, users access rights are not checked in the event of an access attempt. Instead after a successful authentication of the user, all necessary firewall rules should be deployed on all related firewalls in the network at once. As already mentioned, permission is the aggregation of a service, a resource, and all required and forbidden credentials. Therefore an efficient algorithm to find out all enabled permissions for a user, is needed. Let Z the set of all zones exiting in the Policy Framework. Let C_{ZN} the list of credentials, which are needed to access a zone. For Zone z, that would be C_{ZNz}. Let C_{ZFz} the list of forbidden credentials. C_{PNz} and C_{PFz} are of the same kind but are belonging to permissions. Let C_U the set of credentials the user owns. Let Z_A the set, which includes all zones the user is allowed to access. It is defined as follows:

$$Z_A = \{z \in Z | ((C_{ZNz} \subseteq C_U) \wedge (C_U \bigcap C_{ZFz} = \emptyset))\} \quad (4)$$

The user is allowed to access a zone if the needed credentials to access the zone are a subset of the attributes of the user. Also the user must not have a credential which is forbidden. Due to the nature of inheritance of the zones, follows that no one can access resources from a zone, which inherits from a zone on which the user is not allowed to access. Thus, not all zones have to be searched. Because if a zone that the user is not allowed to access is found traversing a branch, you can break up and go on with the next branch. After figuring out the zones, which the user is allowed to access, all permissions inside these zones, enabled for the user, have to be evaluated. So one have to check all permissions, if the user has all needed credentials, which are necessary for permission. In addition the user must not have an attribute, which is forbidden for this permission. This results in:

$$P_A = \forall_{z \in Z_A} \{p \in z | ((C_{PNz} \subseteq C_U) \wedge (C_U \bigcap C_{PFz} = \emptyset))\} \quad (5)$$

The result is a set permission which can be converted into firewall rules. Since this contains only rules that allow connections and not forbid them, the order of the rules is irrelevant. So, there can not be any conflicts between those rules.

IV. Related Work

Dynamic control of firewalls has been studied in detail for Voice over IP applications [12], [6], [13]. Cisco Systems's TrustSec technology [14] can deploy "downloadable Access Control Lists" (dACL) when a user connects to the network. However, this is currently a vendor-specific solution.

Bartal et al. [15] present a firewall management toolkit with includes a management language for abstract firewall rules, but every change of the rule set result in a recompilation of the whole model code. Due to the distributed structure, this could become a bottleneck in DynFire.

Frèdèric Cuppens et al. [16] proposed a framework for describing rules for different firewalls, but the solution is more focused on the description of a firewall rule. This framework also does not regard the group or user based access control. Instead, they grant access on a per host basis.

Laborde et al. [17] used a general RBAC model to design a Policy-based network management (PBNM) system, but kept open if it is possible to implement it into a real network due to its complexity.

Basile et al. [18] designed and evaluated an ontology-based security policy system for networks. They regard the administrative task but also do not regard the dynamic login of different users. They also assumed that a user uses a fixed workstation and so the user could not change the location.

Zhang et al. [19] defined a high-level specification language. Their approach detects conflicts in the rule specification but can not automatically deal with it. The Policy Framework we propose does not allow the specification of deny rules and completely avoid any conflicts in the rule generation.

XACML [20] is a very detailed policy description language and was investigated during the specification phase. The specifications of XACML is not powerful enough to fit the requirements, e.g., describing the distributed administration. An overview about the complete DynFire architecture is given in [21]. This also covers the other developed modules.

V. Conclusion and Future Work

ABAC provides a solid basis for implementing an attribute-based policy framework. Nevertheless, a realization of such a framework imposes high complexity in the evaluation of the policies and the administration. This paper presented the Policy Framework of DynFire, an ABAC-based framework for the dynamic modeling and deploying of firewalls. It enables attribute-based access to network resources.

We currently deploy DynFire in the campus network of the University of Stuttgart. While finishing the implementation, we are also evaluating and analyzing the performance, scalability, and security of DynFire.

Acknowledgment

This work was supported by the “DynFire” project [22], a research project supported by the German Federal Ministry of Education and Research (BMBF, Foerderkennzeichen: 01BY1151). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DynFire project or the BMBF.

The authors would like to thank Christoph Schock and Dominik Lamp for contributions and feedback.

References

[1] R. Oppliger, “Internet security: firewalls and beyond,” *Commun. ACM*, vol. 40, no. 5, pp. 92–102, May 1997.

[2] S. M. Bellovin and W. R. Cheswick, “Network firewalls,” *IEEE Communications Magazine*, vol. 32, pp. 50–57, 1994.

[3] D. B. Chapman and E. D. Zwicky, *Building Internet Firewalls*, 1st ed., D. Russell, Ed. Sebastopol, CA, USA: O’Reilly & Associates, Inc., 1995.

[4] IEEE LAN/MAN Standards Committee, “Station and Media Access Control Connectivity Discovery,” IEEE, Std. 802.1ab, 2009.

[5] J. Schoenwaelder and T. Jeffree, “Simple Network Management Protocol (SNMP) over IEEE 802 Networks,” IETF, RFC 4789, Nov. 2006.

[6] S. Kiesel and M. Scharf, “Modeling and performance evaluation of transport protocols for firewall control,” *Computer Networks*, vol. 51, no. 11, pp. 3232–3251, Aug. 2007.

[7] M. Stiemerling, J. Quittek, and C. Cadar, “NEC’s Simple Middlebox Configuration (SIMCO) Protocol V3.0,” IETF, RFC 4540, May 2006.

[8] “Astaro, a Sophos Company,” http://www.astaro.com, [retrieved: Oct., 2012].

[9] D. F. Ferraiolo and D. R. Kuhn, “Role-based access controls,” *15th National Computer Security Conference*, pp. 554–563, 1992.

[10] J. M. Torsten Priebe, Eduardo Fernandez and GÃ$\frac{1}{4}$ntherPernul, “A pattern system for access control,” in *Research Directions in Data and Applications Security XVIII*, ser. IFIP International Federation for Information Processing, C. Farkas and P. Samarati, Eds. Springer Boston, 2004, vol. 144, pp. 235–249.

[11] IEEE LAN/MAN Standards Committee, “Port-Based Network Access Control,” IEEE, Std. 802.1x, 2004.

[12] C. Aoun, “Plan de signalisation Internet pour l’interfonctionnement entre NAT et Firewall,” PhD Thesis, ENST, Paris, 2005.

[13] ETSI TISPAN, “NGN Functional Architecture,” ETSI, Standard ES 282 001 V3.4.1, 2009.

[14] Cisco Systems, Inc, “Cisco TrustSec Solution Overview,” http://www.cisco.com/en/US/netsol/ns1051/index.html, [retrieved: Oct., 2012].

[15] Y. Bartal, A. Mayer, K. Nissim, and A. Wool, “Firmato: A novel firewall management toolkit,” *ACM Trans. Comput. Syst.*, vol. 22, no. 4, pp. 381–420, Nov. 2004.

[16] T. S. Frèdèric Cuppens, Nora Cuppens-Boulahia and A. Miège, “A formal approach to specify and deploy a network security policy,” in *Formal Aspects in Security and Trust’04*, 2004, pp. 203–218.

[17] F. B. Romain Laborde, Michel Kamel and A. Benzekri, “Implementation of a formal security policy refinement process in wbem architecture,” *J. Netw. Syst. Manage.*, vol. 15, no. 2, pp. 241–266, Jun. 2007.

[18] C. Basile, A. Lioy, S. Scozzi, and M. Vallini, “Ontology-based policy translation,” in *Computational Intelligence in Security for Information Systems*, ser. Advances in Intelligent and Soft Computing, A. Herrero, P. Gastaldo, R. Zunino, and E. Corchado, Eds. Springer Berlin / Heidelberg, 2009, vol. 63, pp. 117–126.

[19] B. Zhang, J. R. Ehab Al-Shaer, Radha Jagadeesan, and C. Pitcher, “Specifications of a high-level conflict-free firewall policy language for multi-domain networks,” in *Proceedings of the 12th ACM symposium on Access control models and technologies*, ser. SACMAT ’07. New York, NY, USA: ACM, 2007, pp. 185–194.

[20] “Xacml spezification - oasis extensible access control mark-up language,” http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, [retrieved: Oct., 2012].

[21] A. Vensmer and S. Kiesel, “Dynfire: Dynamic firewalling in heterogeneous environments,” in *Internet Security (WorldCIS), 2012 World Congress on*, June 2012, pp. 57 –58.

[22] “DynFire project home page,” http://www.dynfire.org, [retrieved: Oct., 2012].