Security Challenges with Cross-Domain Information Exchange: Integrity and Guessing Attacks

Paal E. Engelstad, *Senior Member, IEEE,*

***Abstract*—Current research on cross-domain information exchange is advocating to move away from the inflexible Bell-La Padula (BLP) model, into a more complex policy-driven security model where information objects and end-users are characterized in terms of complex meta-data. It will lead to higher flexibility but will also rely not only on guards, but also on automatic or semi-automatic tools for forming and processing the meta-data. In this paper, we point out some potential pitfalls with this approach. The paper focuses specifically on the relaxation of the BLP security model for confidentiality and discusses security concerns that arise from the use of such tools in combination with guards.**

***Index Terms*—Security, classification, labeling, cross-domain information exchange.**

I. Introduction

During the last couple of decades the NATO allies have gradually realized that the old paradigm of "need-to-know" is limited and inflexible, and might inhibit efficiency of military processes and operations. The need-to-know often results in isolated single-domain information silos.

As a result, a new philosophy of "need-to-share" is starting to emerge [1], [2], [3]. This requires that we try to interconnect the single-domain information silos, without missing the opportunity to protect the information. This is referred to as Cross-Domain Information Exchange. Such information sharing is considered a strategic capability and a basis for obtaining information superiority by ensuring that all allies have the newest and most pertinent information at hand at any time [4]. An example of information sharing might include communication and exchange of information objects between NATO Secret systems, such as the NATO forces, non-NATO partners and unclassified networks (e.g., of collaborating international organizations) [5]. Current research on cross-domain information exchange is advocating to move away from the inflexible Bell-La Padula model [6], with simple security classifications and simple security clearances, into a complex policy-driven security model where information objects and end-users are characterized in terms of complex meta-data. The Content-based Protection and Release (CPR) is an important work within the area, and stands as an example of this evolution [7] This will lead to higher flexibility but will also rely on automatic tools for forming and processing the meta-data. The first part of the paper assumes a system with simple security classifications and clearances. However, the paper demonstrates that the same conclusions made for this simple system, also apply to systems evolved towards the use of more complex meta-data.

Document submitted on 27. April 2015.

P. E. Engelstad is with Oslo and Akershus University College of Applied Sciences (HiOA), Norway, e-mail: paal.engelstad@hioa.no.

This work was partially funded by the University Graduate Center, UNIK.

Cross-domain information exchange requires some kind of information flow control between the two domains. This is usually implemented by placing a guard between the domains [5] [7] [8]. (An example is the "reactive guard" in Figure 1). The guard inspects the information objects that are attempted to be passed from one domain to the other. The guard may permit some information objects to pass without problem. Other information objects, on the other hand, might be assessed as a security risk and may not be released into the other domain, or they may raise alarms, logging, quarantine and/or other countermeasures. The main functionality of a guard is often thought to be confidentiality protection, while the focus on integrity protection might be somewhat lower. This paper demonstrates how tightly inter-related these are, and calls for a stronger focus on integrity protection.

The paper is primarily concerned with the use of two-way guards, i.e. guards that allow information to pass in both directions. The medium assurance XML labeling guard (XLG) [8] represents a simple example of a solution for cross-domain information exchange, while the High Assurance Guard (HAG) [5] represents a more advanced solution. Such two-way guards release information objects from a "high" (secure) domain onto a "low" (less secure) domain, and accepts objects from the "low" domain into the "high" domain.

This paper points out that even the simplest guard solution breaks with the Bell-La Padula (BLP) model [6], and that this feature of guard solutions can easily be exploited. The fundamental security problems (that applies to both the simplest and most advanced guard solutions) are:

1) The security classification process needed for adding a security label to an unlabeled information object requires a "write-down" signal (in terms of a classification decision) from a higher level of security down to a lower level. This "write-down" breaks with BLP.
2) The blocking functionality of the guard can be used - similarly to another covert channel - to read out this binary signal.
3) The security classification process also requires a "read-down" of topics that are explicitly stated to be classified at a lower level. The paper shows that this integrity vulnerability can be exploited to trigger information leakage, again taking advantage of the "write-down" signal

(classification decision) in point 1 above.

Regarding point 1: The process of a adding a security label to an unlabeled information object requires information at a high classification level. For instance, in a two-class classification system with ”Classified” and ”Unclassified” information objects, adding a label to an object is a high-level process that sends a ”write-down” signal from the ”Classified” security level down to the ”Unclassified” level. Even though this is only a binary signal (i.e. ”label it as Classified” vs ”label it as Unclassified”), the ”write-down” breaks with BLP. (The actual ”write-down” signal corresponds to the ”Classification decision” signal illustrated in Figure 1 below.)

Regarding point 2: The blocking functionality of the guard can be used to reveal the ”write-down” information. This paper describes a ”guessing” attack that takes advantage of this vulnerability and ”pumps” the system for Classified information. The attack is also demonstrated in a Proof-of-Concept experiment presented in the end of the paper.

Regarding point 3: An additional problem with the security labeling process is that to be able to classify whether an object is at a high security level (e.g. ”Top Secret”), one also needs information about which topics should be classified at a lower level (e.g. ”Confidential”). That is, if the topic of a document is explicitly noted as ”Confidential” in the system, the information object should not be classified as ”Top Secret”. Thus, there is also a ”read-down” process that takes place during the security labeling process. This paper explains how this integrity vulnerability can be exploited to trigger information leakage, and an integrity attack (”learning attack”) is demonstrated in a Proof-of-Concept experiment in the end of the paper.

Some may argue that in the future one will move away from course-grained security levels (”Top Secret”, ”Secret”, etc.) into a policy-driven model where information objects are labeled with a description of the topic within, and that release decisions in the guard are taken by comparing the label with content in a policy database. However, this paper demonstrates that the attacks are applicable also to this future situation. In this case, the guessing attack can be used to reveal sensitive and/or classified information in the policy database.

Both the guessing attack and the integrity attack only requires clearance at an ”Unclassified” level. To demonstrate the problem with guards, we first assume a simplistic two-way guard that only focuses on confidentiality protection, i.e. it blocks ”Classified” information to leak out of the ”High” domain, while it lets ”Unclassified” information to pass in. We demonstrate that the aforementioned attacks easily can be carried out from the ”Low” domain. The only way to prevent this is to implement quite strict integrity protection in the guard.

The paper shows that with guard technology it is not sufficient to focus on confidentiality: Integrity protection becomes even more important. This is very worrisome, given that the combination of full confidentiality and integrity protection (e.g., the combination of both the BLP and Biba [9] models) is known to construct information silos. Indeed, the whole intention of using guards is to eliminate those silos, i.e. to facilitate inter-domain information exchange and information sharing. The big question is: Will the relaxation of the BLP model that the guard technology represents call for so strict integrity protection that the information exchange will be even more limited than it would be if we stuck with the pure BLP model?

We would like to point out early that we understand the need for relaxation of the BLP model to facilitate cross-domain information exchange. The purpose of this paper is not to point out that a relaxation is necessarily a bad idea, but rather to bring to attention specific security concerns and threat vectors that must be handled properly by the system, both in terms of good formulation of the security policy and possibly by implementation of additional security functions targeting the new security vulnerabilities that might emerge.

II. Background

A. Security Filter and Policy

One of the main components of a guard is a security filter. For instance, a security filter is shown in the ”reactive guard” in Figure 1. The security filter is mainly used to protect the confidentiality of the domain that the information object is released from. It reads the security label of the information object, consults with the security policy, and makes a decision whether or not to release the object.

B. Deployment scenarios of Automatic Security Classification

A guard usually includes - in addition to the security filter - a module that automatically double-checks the classification level of the information object (Figure 1 - Alternative 1). To do so, the module needs to perform a security classification of an information object based on its content. (Today, this is done by ”Dirty-word” lists, but in the future more advanced classification lists and policy-driven methods is expected to emerge.)

The purpose of the module is to double-check that the security label carried by the object is correct, before the original information object with the original label, is passed to the security filter, possibly along with a modified label indicating the result of the double-check. If these labels are conflicting, the security filter will consult with the policy database to reach the correct release decision.

Such a guard is referred to as a ”reactive guard”, because it acts re-actively, making sure that ”Classified” information is not leaked from the ”High” domain inside an ”Unclassified” information object. Thus, it is aimed at mitigating two threats: The first threat is the mislabeling by a sloppy user and/or by a bad-performing security classification plugin (Figure 1 - Alternative 2 and/or 3). Another threat to mitigate is an intentional attempt to send out ”Classified” information within ”Unclassified” information objects, which might be part of another security attack.

In addition to doing reactive security classification in the guards, security classification is usually also taken place when the information object is created and inserted into the domain

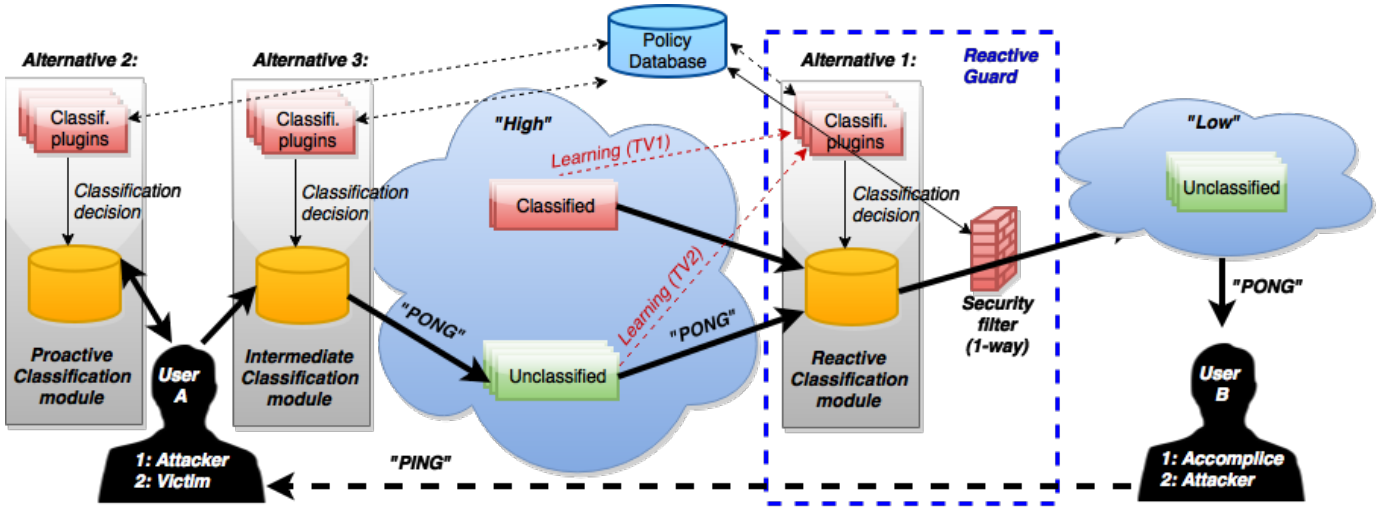

Fig. 1: Example of cross-domain information exchange across a guard. In scenario 1, the attacker is inside the ”High” domain, while in scenario 2, the attacker launches the attack from the outside ”Low” domain.

with a security label. This is referred to as proactive security classification. Typically, this can be done by an author of a document or by a separate reviewer. With the increasing amount of information that needs to be classified and labeled, there is a need for software solutions that are proactively assisting humans in their classification decision (Alternative 2 in Figure 1).

Finally, the classification can also take place in an intermediate node (Figure 1 - Alternative 3). The node might be a labeling gateway that do automatic security classification and (re-)labeling, before the information objects are passed into the domain where the objects are labeled. The intermediate classification can be either reactive (i.e. the information object already carries a label before it reaches the gateway) or proactive (i.e. the information object is unlabeled and obtains its security classification by the gateway).

III. The Guessing Attack

A. Attack on a Reactive Guard (Alternative 1)

In this section we propose a novel attack on a guard that releases information. Consider the same scenario as before (Figure 1 - Alternative 3). For simplicity, we first consider attack scenario 1, where User A is an attacker within the ”High” domain, and User 2 is an ”Accomplice” in the ”Low” domain. We will later discuss Scenario 2, where the attack is launched from the ”Low” domain, and where the attacker is not required to have access to the ”High” domain.)

Assume two collaborating malicious users, the Attacker (User A) and the Accomplice (User B), as illustrated in Figure 1). The Attacker is infiltrating the anti-terrorist organization ”High”, but as a subcontractor he has only clearance for handling ”Unclassified” information in their domain. The ”High” domain of the anti-terrorist organization is connected to the Internet (referred to as the ”Low” domain) where the Accomplice is located. (For illustration, we have assumed that these subjects are persons, but the Attacker could just as well be a piece of malicious software, such as a Trojan, while the Accomplice could just as well be a server.)

It is known by rumors that the organization ”High” has a special Anti-Terror Control Center (ATCC) near the capital city, but the location of the control center is ”Classified”. The terrorist organisation has a list of 16 possible locations, and plans to launch a preemptive strike at the ATCC. For this strike to be effective the organisation needs to know the exact location.

Having only access rights according to the ”Unclassified” clearance level within the ”High” domain, attacker constructs a forged ”Unclassified” information objects containing words like ATCC and 8 of the possible locations on the list, and sends it to the Accomplice (along the ”PONG” route indicated in Figure 1). The combination of the word ”ATCC” and one of the 8 locations, triggers the reactive guard not to release the information object, assuming it contains information about the secret ATCC.

Having narrowed down the guessing list to only 8 locations, the Attacker forms a new similar information object containing the names of four of these locations. This time, the information object is released by the guard and received by the accomplice.

Thus, the Attacker knows that the right location is amongst the 4 locations that were not included in the previously sent information object. The Attacker continues with this binary-search until the final exact location is determined. A practical example of how such guessing attacks can be carried out is demonstrated by the Proof-of-Concept experiments described in Section V.

One may argue that the guard could do something else than blocking the messages from the Attacker. However, if the guard sanitizes the message (removes Classified information) or quarantines the message (delays for human intervention) instead, it will be noticed by the Accomplice, who can

collaborate with the Attacker to find out the exact format of the original message and when it was sent.

B. Guessing attack launched from the "Low" domain

It should be noted that the attacker is not required to have any clearance to construct "Classified" information objects in the attack described above. He is supposedly only creating unclassified information objects that might "accidentally" be labeled - or re-labeled in the reactive guard - as "Classified".

Furthermore, it is not required that the attacker has inside access to the "High" network. Consider for example attack scenario 2 in Figure 1. Now, the Attacker (User B) is outside in the "Low" domain. He communicates by ping-ponging unclassified email messages with an unknowing victim on the inside. The attacker might hide some "garbage" information far down in the lower part of the email; i.e., in the part containing previous email exchanges that are practically never read, and sends it to the victim (indicated by the "PING" route in the figure.) Each time a message is sent by the attacker, the "garbage" part is changed. The unknowing victim will send reply messages back (along the "PONG" route in Figure 1.) If the reply message is not received from the unknowing insider, the attacker knows that the garbage part contained sensitive information.

C. Attacks with Alternative 2 or 3

Security classification that takes place in intermediate nodes (Figure 1 - Alternative 3), is subject to the guessing attack in the same way, whether the classification is reactive or proactive. The only difference is that the security label is set by the intermediate node instead of being set in the classification module on the reactive guard. The blocking functionality of the guard is still used to read out the "Classified" information in the guessing attack. Also in this situation, the attack can be waged from the "Low" domain (cf. Section III-B).

Finally, a user can also launch a guessing attack on the software that assists in proactive security classification (Figure 1 - Alternative 2). Consider a user is getting assistance with the labeling though an automatic software tool after having created an information object (Figure 1 - Alternative 2). The user sends the information object to the software tool. The tool inspects the content, marks the object with a proposed security classification and returns it to the user. Again, a malicious user can wage a "guessing attack" against this software, trying out different text. Unlike the other attacks, the malicious user is strictly required to have access to the "High" domain - and to this particular software, and the covert channel here is not the blocking of the guard but the fact that the user has direct access to knowing which security label is assigned to the security object.

IV. Integrity Attacks and Threat Vectors

A. Integrity Attacks Might be Easier to Launch

A module that performs security classification is essential to protect the site from information leakage. Below, we describe an integrity attack that can manipulate the module to assign a too low classification label (e.g. "Unclassified") to an information object that should have a higher classification (e.g. "Classified").

While the guessing attack compromises confidentiality from correctly rejecting information objects to be released, the integrity attack force the guard to mistakenly leak information objects that should have been blocked. Thus, the attack on the integrity of the system, can also compromise the confidentiality of the domain, however in a different way than the guessing attack.

In some systems, it might be easier to manipulate the integrity of the system, than getting access to confidential information. The reason is that many systems are primarily governed by the BLP model. Confidentiality protection is the main concern, while integrity protection is often of a somewhat lower priority.

B. "Learning attack": Manipulating the Classification Plugin

The module that does security classification must know how to distinguish "Top Secret" topics, from "Secret" topics, and so forth. In a simple world, this knowledge can be manually configured (e.g. by a Dirty Word list"). Moving to a more complex system means also a trend towards the need for automatic learning, i.e. the module will have to learn from existing information object in the system. The classification of the content of new information objects will be decided by comparing it to the learnt knowledge.

Thus, if the set of information objects at a given security classification level in the high domain is manipulated by an integrity attack, the module will give wrong decisions about the security classification of new object, leading to wrong release decisions at the guard. That is, an integrity attack can affect the confidentiality protection of the domain, due to the write-down signal that violates the BLP model. There is a learning threat vector associated with the learning of each classification level in the system. With only two classification levels in the example in Figure 1, the two learning threat vectors is referred to as "TV1" (threat vector 1) and "TV2" (threat vector 2).

For instance, advanced classification modules will use clustering and/or regular machine learning to do classification (e.g. by Support Vector Machines, k-means clustering, etc.). Since such techniques typically make distinctions between for example "Classified" and "Unclassified" topics, the module needs to learn from both the "Classified" and the "Unclassified" objects that exist in the domain, to be able to distinguish one class from the other. This means that the module - which is a "Classified" subject/object/process according to the BLP model - needs to do a "read-down" from the "Unclassified" information objects. The read-down is permitted by BLP, but compromises the integrity of the module.

Thus, by manipulating the "Unclassified" information objects within a domain you can get the system to release "Classified" information objects! An example will be demonstrated by Proof-of-Concept experiments in Section VI. The experiments demonstrates that even though the system operates

entirely according to the BLP model, except from a simple write-down signal from the module, the confidentiality of the entire system can be compromised through the lack of integrity protection of "Unclassified" information. Waging an attack on "Unclassified" information objects, might not be hard in some systems.

V. Guessing Attack Experiments

A. Proof of concept

The intention is to create some simplistic attacks to demonstrate the principles. The attacks are made simple, so they are easy to understand. With simple attacks, it is easy to invent countermeasures. However, it is possible to make the described attacks more advanced, and harder to prevent. The intention of this paper is not to point out, whether the attacker or the defender will win this arms race. The intention is simply to point out that the attacks are possible. In other words, our intention is only to demonstrate the proof of concept.

B. Experimental setting

The experimental setup is shown in Figure 2. An attacker ("User 1") is sending unlabeled documents through a labeling gateway, which assigns the documents with either a "Classified" or an "Unclassified" label. The actual label assigned to a document depends on the content of the document and the functionality of the classification plugin. The attacker wants to send the documents (information objects) to the accomplice ("User 2"). The documents that are labeled as "Classified" are blocked by the guard and not released, while the "Unclassified" documents are released. The attacker and the accomplice collaborate, so that the accomplice informs the attacker about the documents that were received. The attacker can then deduce which documents where labeled as "Classified" by labeling gateway (or the classification plugin) and blocked by the guard.

C. Implementation of the classification plugin

The most crucial part of the experimental architecture is the classification plugin shown in Figure 3. For simply a proof-of-concept experiment, we implement a simple plugin. The plugin has both a "Dirty-Word List" and a classification list. The document is first checked against a manually configured "Dirty-Word list". A Dirty-Word list has a binary outcome; i.e., if the document to be classified contains a word (or expression) that is listed on the "Dirty-Word list", the document is assumed to be "Classified". If not, the document content is then to be checked by the "Classification List" (Figure 3).

The "Classification list" is generated by machine learning. We use all the existing and available documents in the site, that for instance can be assumed to be present in the database (Figure 2). The document is tested against the classification list, and each word in the document that are on the list is assigned a value from the list. From the aggregate value, the plugin calculates p, which is the probability that the document contains "Classified" information. If the probability is higher than 50%, the document is most likely to contain classified information. If not, the document might be sent forward to additional modules for further testing.

The chain of tests proposed in Figure 3, may contains not only the "Dirty Word List" and "Classification list", but any more additional modules, as indicated in the figure. This means that a document must fail all the tests to be considered "Unclassified": If the document has a positive outcome on any of the test, the document will be labeled as "Classified". The reason we propose this structure is to reduce the number of false negatives to a minimum. A false negative means that a document that contains "Classified" information will mistakenly be labeled as "Unclassified". The consequences is information leakage, which can be critical. This proposed decision chain means that the number of false positives will be higher. A false positive means that a document that only contains "Unclassified" information, is labeled as "Classified" and are not released by the guard. The effect of this is inconvenient, but probably much less critical than a false negative. For the purpose of our proof-of-concept experiments, we implemented only the dirty word list and the classification list.

Notice also in the bottom of Figure 3 that we have added an optional topic clustering module that might be implemented in the future. The reason is that we might in the future move towards solutions with policy-driven security architectures. Then, the classification plugin probably will need to determine the different topics of the information object (e.g., by "clustering") and add more advanced topic labels that indicate the topical content of the object. The "topic clustering" process will probably need to consult the policy database to find the correct match between the detected topics and the corresponding labels that should be applied. The fact that the classification plugin

For the actual machine learner that was used to generate the classification list, we apply Lasso (Least Absolute Shrinkage and Selection Operator) [**?**], [**?**]. The reason for using Lasso is that it creates sparse solutions that are easily interpretable by humans [10] [11]. Thus, Lasso is very suitable, because it makes our proof-of-concept experiment easy to analyze.

As for the exact machine learning, we are using a experimental setup that is exactly similar to the two-class benchmark experiment as described in [10] and [11], except for some minor modifications that are explicitly outlined below. Due to space limitations, we are only describing the machine learning process briefly below. Readers should instead refer to [10], [11] and [12] for more detailed information.

D. The information objects

The information objects used in the experiments are documents from the Digital National Security Archive (DNSA) [13]. DNSA contains the most comprehensive collection of historic and declassified US government documents available to the public. From DNSA, we chose US policy documents from three different topics; Afghanistan (AF) in the years 1973-1990, China (CH) in the years 1960-1998 and Philippines (PH) in the years 1965-1986. These were chosen be-

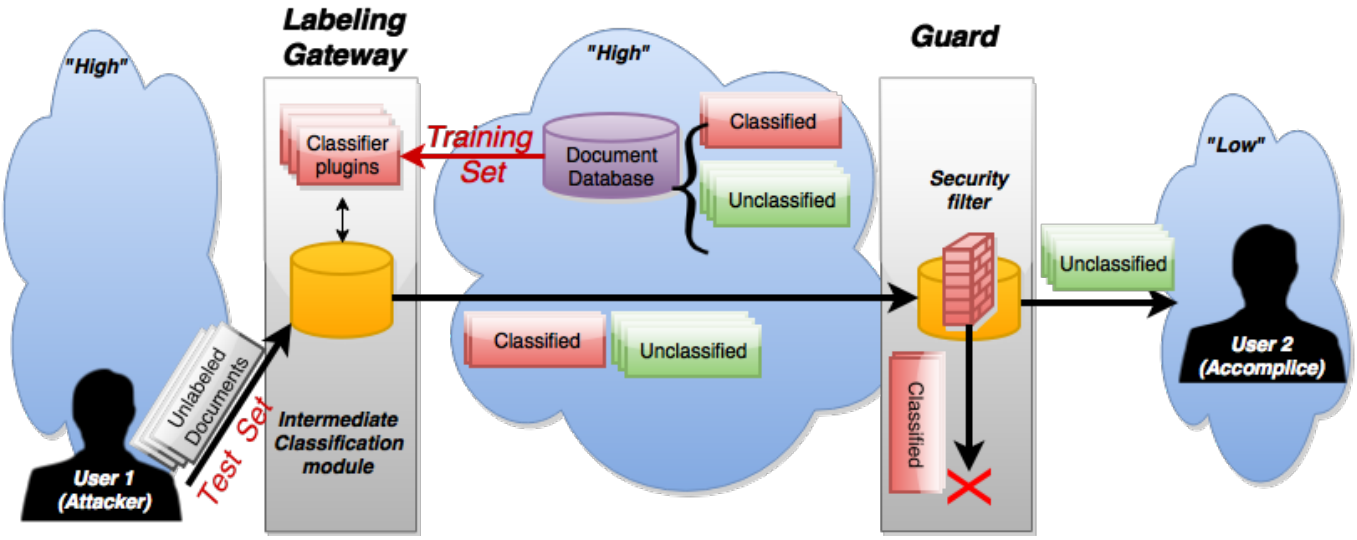

Fig. 2: Experimental setup for proof-of-concept

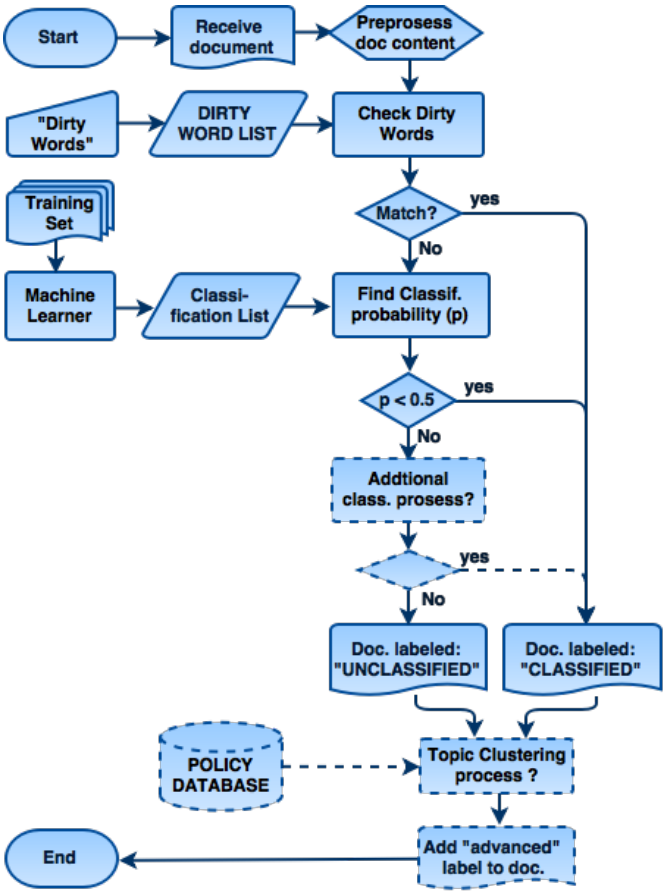

Fig. 3: classification plugin Implementation

cause they contain a mix of both classified and unclassified documents. Of a total of 5867 documents available within these three topics, we removed documents not useful to our experiments (e.g. those with unknown or inappropriate classification [10]), ending up with a total of 2793 documents. Of these, 1070 are ”Unclassified” documents, while the remaining documents are 1155 ”Confidential”, 471 ”Secret” and 97 ”Top Secret” documents. For the guessing attack demonstrated below, we first aggregate the ”Secret” and ”Top Secret” documents into one common ”Classified” class, while we do not use the ”Confidential” documents. The reason is that in many domains, there are less classified than unclassified information. Furthermore, this situation also yields a positive β_0-value of Lasso (cf. [11] and Table I), which makes the guessing attack easier to demonstrate.

Note that for training the classification plugin and generating the classification list, we are only using 70% of the documents. These corresponds to the documents that reside in the Document Database in Figure 2. The remaining 30% of the documents are used as a test set, e.g., we may send it through the labeling gateway to test the classifier performance as indicated in Figure 2.

E. Configuring the classification plugin

The documents of in training set was stemmed (i.e. each word was replaced by its word stem) and stop-words were removed. Each document was treated as a bag-of-words, where the frequency of each word in the document was collected. Looking at all words (or more correctly ”stems”) in the training set, only 1% of the words, namely those with the highest Information Gain (IG), were used in the machine learning. The reason for this massive feature selection is that we want to generate a very short classification list that is easy to analyze in this proof-of-concept study. The documents were fed into a Lasso machine learner, which generated the classification list.

Generally, Lasso is a linear regression model that associates a β-value to each word, and finds the optimal values of all the β-values, by computing the following minimization:

$$\widehat{\beta} = \arg\min_{\beta}\{-\sum_{i=1}^{n}\ln(1+\exp(-\beta^T x_i)) + \lambda\sum_{j=1}^{d}|\beta_j|\}. \quad (1)$$

Here, $\widehat{\beta}$ is the parameter estimates, while x_i is a vector associated with document number i. Each element in the vector corresponds to one word stem in the model, and the value of the element represents the number of occurrences of that word stem in the document.

Due to the L1-norm in the last part of equation 1, Lasso tends to generate sparse solutions. Thus, the initial number of 50 words presented to Lasso after the IG-based feature selection, is reduced further down to a total number of 37 words. Both this machine learning technique and the feature selection mentioned above are described in further detail in [10] and [11].

Lasso is based on logistic regression. After having trained the classifier (i.e. generated a classification list), a document k that needs to get its classification level determined can be compared with the classification list. The number of occurrences in the document of each word that is also found on the classification list is noted down, and the vector x_k for document k is formed. Finally, this is fed back into the logistic function, to calculate the conditional probability that the document belongs to one of the two classes:

$$p(y_i = +1|\beta, x_i) = \frac{1}{1+exp(-\beta^T x_i))}. \quad (2)$$

The β-values are the ones that were determined during the training (i.e. equation 1).The values $y_i \in \{-1, +1\}$ in the equation are class labels that indicate membership to the class "Unclassified" ("+1") or "Classified" ("-1"). Thus, we have defined the class labels such that $p(y_i = +1|\beta, x_i)$ is the probability of the document belonging to the "Unclassified" class.

Finally, we also consider that the organization is running two new top secret projects called "FOENIX" and "NSQ", which have no occurences in the training set. Thus, a dirty word list containing the two terms *foenix* and *nsq* is generated manually and added to the classification plugin, in addition to the classification list that is already generated (Figure 2).

F. Guessing attack on the classification list

On behalf of "User 1" in Figure 2, we first generated a list of words. There are many ways to generate a guessing list. One way is to manually set up short list of selected words. A second way is to download an English dictionary, or a subset of it. A third method is to scan and extract the words from a large number of documents, preferably within a topic related to the contents of the documents in the domain. For simplicity, we chose the latter, and started off with a list of more than 20.000 words.

For each word on the list, we generated a document containing 100 words repeatedly of the word in question, and sent the forged unlabeled document through the labeling gateway on behalf of "User 1".

Since the classification list for simplicity is so short, only 22 documents of the documents that were sent were labeled as "Classified" by the classification plugin and thus not released by the guard. The documents that is blocked corresponds to the the 22 guessed word stems: *ambassadori, clear, intellig, kuan, liaison, like, lord, memorandum, might, moscow, peke, posit, possibl, postur, probabl, relationship, seem, sensit, should, suggest, teng, view* and *warsaw*. This means that the classification module considers documents containing such words as an indication of sensitive content and an indication that the document might contain "Classified" information.

If you think some of the listed words are far-fetched in terms of classification, remember that for the sake of simplicity, we first extracted only around 50 words that Lasso was allowed to use in the first place. Those 50 words were not selected based on their relevance to security issues, by merely by the objective information gain they represent. Out of the total of these 50 words, Lasso used 37 as indicators of security classification, while the remaining 13 words were deemed by the Lasso algorithm to be insignificant for the purpose of classification.(The 13 security-neutral words are *ambassador, believ, chines, chou, communiqu, could, eye, henri, interpret, next, premier, side and wish*.)

To analyze the result of the attack, we have to inspect the entire classification list, which is shown i Table I. We observe that our guessing attack has detected the 22 most sensitive words on the classification list (where*warsaw* is the most sensitive word, since it has the lowest β-value). The only two words not detected by our guessing attack are the two least sensitive words with a negative β-value, namely *chiao* and *soviet*.

G. Merging the dirty-word list into the classification list

In the lower pane of Table I we also find the two terms *foenix* and *nsq*. These terms are given a practically infinitely low β-value. Documents can be evaluated in the same way as for the classification list, i.e. using equation . This means that even for the longest documents, only one occurrence of either the term *foenix* or *nsq* will result in the document to be labeled as "Classified". In this way, it is in fact possible to merge the automatically generated classification list with the manually configured dirty-word list. The net result will be the same as for the classification plugin architecture outlined in 3.

H. A guessing attack on the Dirty Word list

Assume the attacker wants to find any Attacker three-letter acronyms used by the organization, and only those that are not a three-letter stem of a word in a dictionary. First, the attacker generates all combinations of three-letter acronyms, i.e. 17576 acronyms in total, with the 26 letters of the English language. The attacker uses his dictionary to eliminate all three-letter combinations that correspond to a word stem of any word in the dictionary. There are a number of stemming tools available for stemming words. Of the 23477 words in total in our limited

TABLE I: Classification list. Words in the left pane has negative β-values and contribute towards a security classification of "Classified", while words in the right pane has positive β-values and contribute towards a security classification of "Unclassified".

Word stem	β^-	$p_w(w=100)$	Word stem	β^+	$p_w(w=100)$
warsaw	-5,43	(4,77E-24)	talk	0,07	(0,787)
kuan	-4,96	(5,45E-22)	fund	0,15	(0,895)
ambassadori	-3,02	(1,49E-13)	manila	0,20	(0,936)
peke	-2,86	(7,40E-13)	crimin	0,37	(0,987)
sensit	-1,92	(8,57E-09)	question	0,40	(0,990)
intellig	-1,85	(1,82E-08)	china	0,41	(0,992)
lord	-1,31	(3,89E-06)	safeti	0,80	(1-1.71E-04)
relationship	-1,06	(4,79E-05)	human	1,08	(1-1.09E-05)
seem	-1,03	(6,26E-05)	servic	1,31	(1-1.00E-06)
possibl	-1,02	(6,78E-05)	info	1,38	(1-5,00E-07)
probabl	-0,99	(9,73E-05)	refuge	1,98	(1,000)
might	-0,79	(6,62E-04)	dear	6,01	(1,000)
postur	-0,70	(1,63E-03)			
memorandum	-0,69	(1,98E-03)			
liaison	-0,50	(0,013)			
should	-0,45	(0,021)	BIAS:		
posit	-0,24	(0,143)	β_0	0,63	(0,652)
suggest	-0,14	(0,320)			
like	-0,12	(0,353)			
teng	-0,11	(0,379)			
view	-0,09	(0,444)			
clear	-0,09	(0,423)			
moscow	-0,07	(0,481)			
chiao	-0,05	(0,534)			
soviet	-0,02	(0,603)			
foenix	-1E99	(0)			
nsq	-1E99	(0)			

dictionary, we had only 739 stems containing exactly three letters. Thus, after removing these stems, we ended up with 16837 acronyms in total that are shuffled in a random order.

On behalf of the attacker, we split the 16837 acronyms in two, with 8418 acronyms in one part and 8419 in the other. According to our algorithm, we send (on behalf of the attacker) a document containing the smallest number of acronyms of the two parts. In our random run, the acronym ("nsq) was not in this part, and the document was therefore passed to the accomplice. Thus, we split the remaining 8419 acronyms that were not sent in that document in two, and send out a new document containing half of them, that is 4209 acronyms. This time, the document is blocked, which means that the secret acronym ("nsq") turned out to be contained in the sent document. The 4209 acronyms there are split in two and a document containing half of them, namely 2104 acronyms, are sent out. In this way, we use binary search to narrow down. After between 14 - 16 sent documents, we find that the secret acronym is "NSQ".

I. Guessing attack launched from the "Low" domain

It should be noted that in the attacks demonstrated above, the attacker is not required to have any clearance to construct "Classified" information objects. Furthermore, it is not required that the attacker has inside access to the "High" network. As pointed out in Section III-B, such an attack can also be launched from the "Low" domain, unless sufficient integrity protection mechanisms are implemented in the guard.

The aim of the experiments is to demonstrate that the relaxation of BLP creates problem unless integrity protection is introduced to counter-balance the negative effects of the relaxation. Thus, in the experiments it is assumed that the guard only focuses on confidentiality, and not on integrity protection. Thus, the guard is effectively a two-way guard were "Unclassified" information objects can pass freely from the "Low" domain into the "High" domain. Due to this assumption, the guessing attack can easily be launched by an outsider that has no direct access to the "High" domain.

J. Guessing attack on a high-performance solution

In the experiments above, we removed the security-sensitive keywords to create a scenario with a challenging corpus. Now, we launch the same attack on high-performing domain, i.e. on a solution where these keywords are allowed to be taken into account. This time, the keyword *secret* were amongst the sensitive words that were guessed, and the word is the second strongest indicator of a "Classified" document. Listed in increasing order of importance, the detected words were *peke*, *might*, *memorandum*, *henri*, *secret* and *warsaw*

The less important sensitive words that were not detected (i.e.IntegrityAttackPerformance those that are weaker indicators of the "Classified" class than those that were detected) are: *soviet*, *ambassador*, *seem*, *possibl*, *kuan*, *relationship* and *should* (listed in incresing order of importance).

VI. Integrity Attack Experiments

A. Attack by changing or forging "Unclassified" documents

Now, we consider an attacker sending bogus "Unclassified" information into the "High" domain, e.g. by communicating with an unknowing victim on the inside. Either, the attacker can generate new documents that are forged as real "Unclassified" documents. Alternatively, he may change existing ones, if the integrity protection of "Unclassified" information objects is not high. For simplicity, we assume the latter in the following. However, conclusions will be the same.

Assume that the attacker manipulates a number of "Unclassified" documents, so that each contains w words of each of the sensitive words detected in the guessing attack.

An attacker can often easily guess some sensitive words. In the demonstration presented here, we assume that the attacker has used the guessing attack to obtained the same sensitive words that were detected in the guessing attacks presented above, and that the attacker adds w of each of these words into the "Unclassified" documents that he is able to manipulate.

We assume first that the attacker attacks the high-performing system described above, i.e. as described in Section V-J. He forges documents containing 100 words of each of the detected sensitive words (i.e., $w = 100$), using the sensitive words that were detected in the guessing attack above on that specific system (cf. Section V-J).

B. Demonstration of the impact of the attack

By communication with the domain (e.g. using the ping-ponging described above), such "Unclassified" information

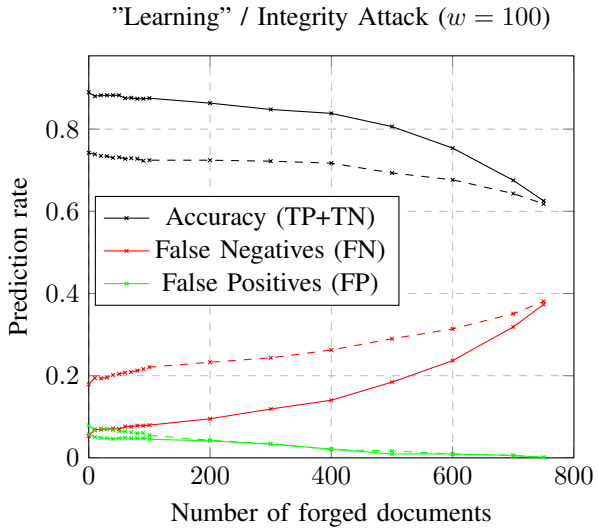

Fig. 4: The impact of "learning attack" on performance. Red lines are the rates of the false negatives, where solid vs dashed lines represent attack on a high vs a low performing system (i.e., keywords included vs ignored.

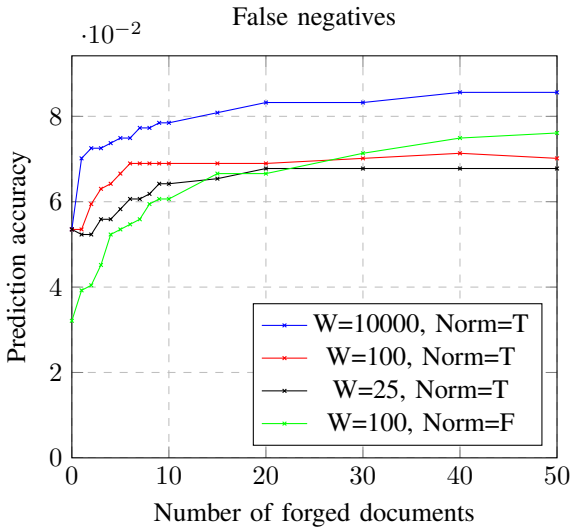

Fig. 5: The impact of "learning attack" on the false negatives in a high performing system (keywords included)

might easily be stored in the "High" domain. Stored documents are typically used later for the training, since keeping the machine learner up-to-date is a requirement to maintain performance. The attack has impact on the training, because the manipulated "Unclassified" documents are now included in the training set used to generate the classification list in the classification plugin. In this way, the attacker changes the performance of the output of the classification plugin. How much the performance is manipulated, depends on how many of the "Unclassified" documents used in the training that the attacker is able to manipulate.

In this part, we have only assumed that the attacker is able to inject "Unclassified" information into the "High" domain. Thus, we are only studying the impact of the attack on the training of our machine learner, as a function of the number of manipulated documents. Results are shown in Figure 4.

We are mostly concerned about the attacks impact on the False Negative (FN) rate, shown by the red curves in Figure 4. The reason is that a false negative means that the classifier assigns "Unclassified" to an information object that "should be" classified as "Classified". This means that the document can be mistakenly leaked out of the domain, and the confidentiality of the domain is compromised.

The false negative rate is measured by the test set. Since the test set (which is only used for our analysis) is unchanged by the attack, we compare the real, actual classification of each document in the test set with the classification level that the classifier would assign to it. The solid red curve in Figure 4 refers to the high-performance domain where the rate of false negatives is low in the first place (cf. Section V-J). We observe that the curve is steadily increasing. This means that for a hign number of manipulated documents, the attacker is gradually destroying the training set, and the information leakage will easily increase correspondingly.

Now, assume that the attacker launches a similar attack on the low-performance domain (cf. Section V-F), meaning that the guessed words used in the manipulated documents are also replaced with those detected in Section V-F. The resulting False Negative rate is shown by the dashed red curve in Figure 4. The curve starts off by a higher False Negative rate, since this correponds to the performance of the low-performance domain before the attack. Otherwise, the curve follows the same pattern. Both curves meet at the same point when the number of manipulated documents reaches 751. This is not unexpected, as the total number of "Unclassified" documents in the training set is 751.

More interestingly is perhaps what happens at a smaller scale in Figure 4 where the number of manipulated documents is typically less than 10. We observe that the rate of the False Negatives are raising sharply, before it quickly stabilizes. The sharp increase in this part can be explained by the insertion of the sensitive words into the "Unclassified" training set. This is in contrast to the gradual increase for higher number of manipulated documents discussed above, where much of the slow increase is attributed by the fact that the entire training set is getting corrupted.

The small-scale effect of only a few documents manipulated is shown at a higher resolution in Figure 5. To give a simple overview, the Figure shows only results for the attack on the high-performance domain. The solid red curve in Figure 5 is the same as the solid red curve in Figure 4 (except for the differences in scale on the x-axis). We observe a shart increase in the rate of False Negatives, which means that the attack is quite efficient after being able to manipulate only a few documents.

So far we have only considered a scenario with $w = 100$. As

expected, if the attacker inserts a higher number of sensitive words in the manipulated documents, the impact of that document on the training gets higher, and the attack gets more effective. This is shown by the three curves labeled with "Norm=T". ("Norm-T means that for these curves, we normalized the word frequencies by the square root of the document lengths, following the same procedure in our previous works [10], [11] and [12].) However, the effect of adding more words in each document is also decreasing with an increasing number of inserted words. Indeed, using only a modest number of words, e.g., $w = 25$, has also a quite significant impact on the system.

The green curve shows that without normalization wrt document lengths ("Norm=F"), the rate of false negatives is better. On the other hand, document normalization might mitigate the negative effect of the integrity attack. A further study on the effects of various parts of training algorithms is left for future work. The study should focus particularly on the preprocessing steps to the training (e.g., normalization) and how different steps impact on false negatives, attack mitigation, etc.

VII. Conclusion

Current research on cross-domain information exchange is advocating to move away from the simple, course-grained security classifications and clearances, into a more flexible, fine-grained policy-driven security model where information objects and end-users are characterized in terms of complex meta-data. Both policy databases, guards and (semi-)automatic tools - such as topic classifiers, policy databases or security classifiers - are needed to realize this vision. The increased automation and flexibility comes at a cost of a relaxation of the Bell-La Padula model. The paper shows that this leads to concrete vulnerabilities that might be easy to exploit - especially in combination with the use of guards.

First, many of the required tools (e.g. a security classifier or a policy database) will need to have some knownledge about what information is more sensitive and what information is less sensitive, and this knowledge is sensitive information in itself. The "guessing attack" is designed to obtain this knowledge, by the use of the blocking functionality of the guards.

The drive towards more complexity, means a higher need for tools that automatically learns from the environment. The "learning" attack is an integrity attack on this learning process that leads to information leakage at the guards of potentially all possible information objects within the domain.

Both the "guessing attack" and the "learning attack" are demonstrated by proof-of-concept experiments. To be easy to grasp and analyze they were shown in a simplified context, e.g. in terms of a traditional system with two security levels. Still, a more complex model (or at least part of it) will still have topics that will form a lattice in terms of different levels of security. Therefore, the issues with "write-down" will still persist; and the BLP-based approach to analyse the threats will still be valid. Thus, the fundamental security concerns presented in this paper are not eliminated by moving into a more complex model. The biggest difference is that in a more complex policy-driven model, the guessing attack described in this paper will not primarily reveal the content of a secret classification list, but will instead pull sensitive information directly out of the policy database.

The paper demonstrates that the relaxation of BLP will open up for a new family of integrity attacks and threat vectors that in the next step also can lead to compromised confidentiality. This means that it is not longer sufficient to focus on confidentiality: Integrity protection becomes even more important. This is very worrisome, given that the combination of full confidentiality and integrity protection (e.g. the combination of both BLP and Biba) is known to construct information silos. So, will the envisioned relaxation of the BLP model call for so strict integrity protection, that the information exchange becomes even more strict than the pure BLP model?

The work in this paper does not represent a show-stopper for flexible cross-domain information exchange. However, it points out issues that need to be considered in future work.

References

[1] C. Cavas, *Petraeus: US must share more info with allies*. [Online], http://www.defensenews.com/story.php?i=4623591, 2015. [Online]. Available: http://www.defensenews.com/story.php?i=4623591

[2] P.-P. Meiler and M. Schmeing, *Secure Service Oriented Architectures (SOA) supporting NEC, (Technical Report TR-IST-061) NATO*, 2009.

[3] U. Wolf, "Does NATO meet the challenge of the information era?" in *23rd International Workshop on Global Security, Berlin, Germany*, 2006.

[4] N. Brown, *Statement for the Record Before the 108th Congress Committee on Armed Services*. US House of Representatives, 2003.

[5] K. Wrona and G. Hallingstad, "Development of high assurance guards for nato," in *Proc. Military Communications and Information Systems Conference (MCC)*, 2012.

[6] D. E. Bell, "Looking back at the bell-la padula model," in *Proc. 21st Annual Computer Security Applications Conference (ACSAC)*, 2005.

[7] K. Wrona and S. Oudkerk, "Content-based protection and release architecture for future nato networks," in *Proc. Military Communications Conference*, 2013, pp. 206–213.

[8] K. Wrona, S. Oudkerk, and G. Hallingstad, "Development of high assurance guards for nato," in *Proc. Military Communications Conference*, 2010.

[9] K. Biba, "Integrity considerations for secure computer systems," MITRE Tecnical Report, MTR-3153, 1977.

[10] P. E. Engelstad *et al.*, "Automatic security classification with lasso," *Proceedings of The 16th International Workshop on Information Security Applications (WISA 2015), Jeju Island, Korea, August 20-22*, 2015.

[11] P. E. Engelstad, H. L. Hammer, A. Yazidi, and A. Bai, "Advanced classification lists (dirty word lists) for automatic security classification," *Proceedings of The 7th IEEE International Conference on Cyber-enabled distributed computing and knowledge discovery (CyberC, 2015), Cyber Security and Privacy (CSP), Xian, China, Sept 17-19*, 2015.

[12] ——, "Analysis of time-dependencies in automatic security classification," *Proceedings of The 7th IEEE International Conference on Cyber-enabled distributed computing and knowledge discovery (CyberC, 2015), Cyber Security and Privacy (CSP), Xian, China, Sept 17-19*, 2015.

[13] Digitial nation security archive. "http://nsarchive.chadwyck.com/home.do". Accessed: 2015-03-26.