Innovations Syst Softw Eng (2016) 12:81–94
DOI 10.1007/s11334-015-0261-7

ORIGINAL PAPER

Metrics for V&V of cyber defenses

Martin S. Feather[1] · **Joel M. Wilf**[1] · **Joseph Priest**[1]

Received: 11 February 2015 / Accepted: 7 October 2015 / Published online: 26 October 2015

Abstract There is a need for a disciplined approach for evaluating a cyber defense prior to its introduction into an operational environment. This is necessary to assess whether the benefits of the defense will be worth its costs and risks. A traditional V&V workflow is adapted for this purpose. The considerations it must take into account are described, as is the collection and presentation of pertinent metrics. An example of this workflow is given for a cyber defense against a "reconnaissance attack" that threatens information integrity and confidentiality.

Keywords Cybersecurity · Verification and validation (V&V) · Metrics · Testbed · Fidelity · Visualization

1 Introduction

The decision of whether or not to introduce a cyber defense into an operational environment hinges on comparing its benefits to its costs and risks. It is often too dangerous for this decision to be made by first deploying the defense into the actual operating environment and taking measures of its efficacy, while subjecting that environment to deliberate cyber attacks. Instead the decision must be informed from testing of the defense in a "sandboxed" environment, where experiments can be run without fear of disrupting operations or of inadvertently releasing malware. This suggests an analogy with the V&V process for testing of new systems or technologies. As with ordinary systems V&V, there is a concern that testing cannot necessarily recreate the full fidelity of the operational conditions.

Section 2 describes the major factors that have to be taken into account when assessing a cyber defense to be able to make an informed decision about its introduction. These are its benefits (how effectively it performs its intended defensive purpose, plus additional benefits if any), costs (which include monetary costs, computation resources, and potential inconveniences introduced by the defense) and risks (new or exacerbated vulnerabilities from the defense itself, and circumstances in which minor inconveniences could prove to be serious).

Section 3 discusses the issue of fidelity as it pertains to evaluation of cyber defenses.

Section 4 presents the V&V workflow we have devised to approach the evaluation and assessment of cyber defenses. Briefly, its steps are:

Set Up	Identify the cyber attack under consideration, and configure the test environment to be able to execute the attack and explore potential defenses.
Attack	Conduct the attack in the test environment without the defense present and measure its characteristics.
Defend	Select and/or design and implement the proposed defense to the attack, and measure its characteristics when operating in the test environment.
Verify	Scrutinize the results from runs in the test environment to extrapolate to what would happen in the real environment; assess whether or not to go ahead with experiments in the real environment (or if not, to guide revisions to the defense).

✉ Martin S. Feather
martin.s.feather@jpl.nasa.gov

Joel M. Wilf
Joel.M.Wilf@jpl.nasa.gov

Joseph Priest
jjpriest25@gmail.com

[1] Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Validate Under carefully controlled conditions, field the defense in the real environment and (perhaps) conduct the attack in that real environment to measure the effectiveness of the defense. Use the results to inform the decision of whether or not to permanently deploy the defense. Also use the results to assess the fidelity of the extrapolation from experiments in the test environment, scrutinizing fidelity discrepancies to inform the improvement of future conduct of the workflow.

Deploy Deploy the defense to become a normal part of the operational environment (perhaps at first in a "probationary" period).

Section 5 reports on our experience following the workflow through several of its steps, using a proposed defense to a "reconnaissance attack" for illustration.

2 Areas of concern when introducing cyber defenses

When contemplating the introduction of a cyber defense capability, the primary areas of concern are its costs and benefits, and its potential for introducing or exacerbating risks. Each area has to be assessed to gauge the acceptability of the cyber defense.

2.1 Costs

The costs of a defense can involve some or all of the following factors:

Budgetary for example, costs of purchases and licenses of defense software, labor costs associated with installing and maintaining defense software, labor costs of operating the defense (e.g., a helpdesk), and trainer and trainee costs of mastering the defenses.

Computational resources the defense's consumption of computational resources (CPU time, memory, file space and bandwidth). The acceptability of its consumption of these resources will hinge on the adequacy of unused capacity of those resources in the operating environment.

User inconvenience examples include extra steps imposed on the users by the cyber defense, decrease in usability, interference (e.g., from false positives), and curtailed capabilities.

2.2 Benefits

The benefits of the defense cover the ways the defense assists in protecting against cyber attacks, and any additional benefits that might accrue; these can involve some or all of the following factors:

Nature of defense its categorization as a prevention (inhibits a step of a cyber attack), detection or recovery (e.g., backup of critical data so as to be able to restore it should an attack have corrupted the original). For detections, also categorize the nature of its response (e.g., whether detection triggers a notification of a user or system administrator who then decides whether and how to respond, or whether the detection triggers an automated—and thus presumably speedy—response). Also of potential importance is the extent to which the defense logs capture details of an attack, which can be used for subsequent forensic investigations, or for assessing the landscape of possible attacks (e.g., attempts at port scanning).

Additional security (if any) gained by the defense namely while specifically designed to address one kind of attack, the degree to which the defense addresses others.

Efficacy of the defense its probabilities of acting as intended (sensitivity and specificity—i.e., its propensity for avoiding false positives and false negatives); for detections, its responsiveness (how long or how far an attack is allowed to progress before it is detected and a response performed).

Additional benefits (if any) that might accrue from the defense for example, consider a defense designed to delete "orphaned" processes to terminate unauthorized access; such a defense might also have the helpful side effect of terminating orphan user processes that serve no purpose, but which consume system resources.

2.3 Risks

The risks of the defense are those newly introduced or exacerbated by the defense, for example:

Vulnerabilities the potential for new or increased vulnerabilities that an attacker might exploit have to be identified and assessed. For example, granting an automated defense the capacity to kill a user process adds the vulnerability of an attacker subverting the defense itself to cause harm. A defense that impeded or undermined another defense would also be of concern.

Critical interference in addition to inconvenience to routine user operation, considered as part of the costs of the defense, it is also important to assess the potential for loss of *critical* user capabilities. For example, during unusually time-critical circumstances, what would otherwise be just a minor inconvenience to a user (e.g., to have an action separately authorized by a systems administrator) could be a major impediment to a timely response.

3 Assessment and fidelity

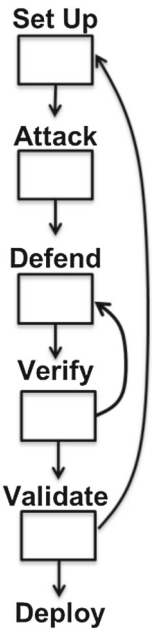
As described in the introduction, we cannot expect to be allowed to assess a cyber defense's costs, benefits and risks by deploying it into the actual operating environment and taking measures of its efficacy, while subjecting that environment to deliberate cyber attacks. Instead we must find other means to seek assurance as to its suitability and efficacy prior to making the deployment decision. Even then, deployment might begin in a probationary period to gauge its performance before committing to permanent deployment.

The approach being followed in the JPL Cyber Research Defense Lab is to develop a test environment in which to safely, securely and repeatably conduct such tests without risk of harming or disrupting the ongoing operational environment. Measures taken of the defense operating in this test environment provide the information we seek. However, this approach raises an important *fidelity* question: to what extent can we gather results in the *test* environment and infer from them what the effects would be in the *operational* environment? Since our concern is with software, at first glance it might seem we should be able to follow the "Test like you fly, fly like you test" principle (e.g., cited as a Lessons Learned in [5]) and recreate the exact environment. However, this is infeasible for the following reasons:

- The operational environment will have more computational resources (e.g., workstations, routers, storage devices) than the test environment can afford to duplicate. While virtualization (e.g., in USC/ISI's DeterLab [9]) can help, it cannot perfectly replicate an operational environment. For example, some malware is able to detect when it is running inside of a virtual environment and changes its behavior [10].
- The work patterns of the extensive user community (e.g., dozens of users) in the operational environment cannot be recreated in the test environment.
- The full scope of computational demands in the operational environment will not be present in the test environment.
- The connectivity between the operational environment to the broader institutional assets and services, and to the Internet, will be absent from the deliberately isolated test environment.
- The operational environment may continue and evolve for years, far beyond the durations that relatively short-lived tests can achieve.

For space systems, it is often impossible to comprehensively recreate their ultimate operational conditions (e.g., outside of Earth's gravity well); as stated in [3] "The lunar environment cannot be sufficiently emulated on Earth, therefore system verification testing will rely to some extent on extension by analysis and ultimate testing in the field (lunar operations)." We take a similar approach, one that runs tests in an environment we can create coupled with analysis and extrapolation of its results, and ultimately (having gained sufficient confidence to do so) fielding in the operational environment.

To approach this systematically, we have devised a workflow that covers the setup and use of the test environment, with special focus on the fidelity mismatch between test and real environments. Our aim is to be able to develop confidence that we can extrapolate from the observations made in the test environment to the costs, benefits and risks to be had in the real environment. Our workflow is described next.



4 A V&V workflow for the assured deployment of cyber defenses

Our workflow for testing of cyber defenses is modeled after a traditional V&V testing workflow, but with an emphasis on recreating conditions pertinent to cyber attack and defense. This workflow is shown in the figure to the right. The boxes represent its steps, the downward arrows indicate progression from a successful outcome of one step to the next, and the upward arrows indicate an unsuccessful outcome of one step leading to the need to return to an earlier step.

This work assumes that a determination has already been made of the driving security needs (a mix of confidentiality, integrity and availability), and potential vulnerabilities that threaten those needs have already been identified.

The steps are:

- Set up—identify the cyber attack under consideration, and configure the test environment to be able to execute the attack and explore potential defenses.
- Attack—conduct the cyber attack in the test environment without the defense present and measure its characteristics.
- Defend—select and/or design and implement the proposed defense to the attack, and measure its characteristics when operating in the test environment.
- Verify—scrutinize the results from runs in the test environment to extrapolate to what would happen in the real environment; assess whether or not to go ahead with experiments in the real environment (or if not, to guide revisions to the defense).
- Validate—under carefully controlled conditions, field the defense in the real environment and (perhaps) conduct the attack in that real environment to measure the effectiveness of the defense. Use the results to inform the decision of whether or not to permanently deploy the defense. Also use the results to assess the fidelity of the extrapolation from experiments in the test environment, scrutinizing fidelity discrepancies to inform the improvement of future conduct of the workflow.
- Deploy—deploy the defense to become a normal part of the operational environment (perhaps at first in a "probationary" period).

These steps are explained in more detail in the subsections that follow.

4.1 Set up

This first step culminates in configuration of the test environment for testing the cyber defense. To do this, it is necessary to:

(a) Identify the system and usage scenario to be defended—"system" refers to the set of computing resources (including users) that operate together, "scenario" refers to the nature of their collaborative work. Identifying these is necessary because different situations will have different security needs (e.g., a design group may have the security needs of integrity and confidentiality, whereas a group operating a system may have the additional and more pressing need of availability). Doing this will involve traditional cybersecurity practices of determining the prevailing confidentiality, integrity and availability needs (e.g., as defined in [6]) of the system in its various usage scenarios. Approaches to assessing the security needs of organizations that operate control systems are widely available (e.g., the Cyber Security Evaluation Tool (CSET®) at https://ics-cert.us-cert.gov/Assessments).

(b) Design or select the cyber attack to be defended against. In keeping with the identification of system and scenario, pick the cyber attack relevant to the security concerns of that scenario (e.g., a denial-of-service attack may be of minor concern to a design group, but of great concern to an operations team).

(c) Determine that the cyber attack would work—through observation, analysis or (if acceptable) testing in the real environment. Efforts to develop cyber defenses should of course be focused on attacks that are feasible. If it is acceptable to do so, the ideal is to try out the attack in the real environment to see if it works. It is likely, however, that this will be unacceptable, in which case it will be necessary to rely upon observation of that attack having taken place in similar circumstances, or on analysis to show its feasibility.

(d) Configure the test environment to model the aspects of the real environment as required for testing. This requires consideration of what aspects of the real environment will need to be replicated, and to what degree, within the test environment. Of key concern is whether the results of testing defenses in the test environment will be indicative of whether those defenses will work in the real environment and what they will cost. Further consideration of this is addressed in the Verify and Validate steps.

4.2 Attack

The purpose of this step is to determine whether the attack succeeds in the test environment without the defense present and active, and gather measurements of the attack.

It is obvious that the effectiveness of a defense can only be demonstrated if the attack would have succeeded if the defense were not in place. This step assures that this is the case before moving on to the next step. Furthermore, this step will be used to gather characteristics of the attack—for example, what kind of security breach was the attacker able to achieve (observation only? observation *and* control?), how extensive, and for what duration?

4.3 Defend

The purpose of this step is to introduce a defense, and measure its operation and efficacy.

Begin by selecting (if there are choices), or designing and implementing a proposed defense, and introducing it into the test environment.

First operate and measure the defense's costs and additional benefits (those other than the benefit of foiling the attack) *without* the attack running—measurements under these conditions are necessary to be able to assess whether the defense is acceptable to deploy. Then perform tests in which the attack is employed, and measure the defense's success at countering the attack.

It is important that the set of tests covers the range of conditions under which the defense will be called upon to operate, so as to be assured that it will be sufficiently effective across that range. As stated in [4] "If the measurement is flawed, or the experiment does not account for all possible mechanisms for affecting the measurement, then any result is not fully justifiable." This may be especially challenging for cyber defenses taking the form of anomaly-based intrusion detection systems that aim to distinguish "normal" behavior from the behavior of an actual cyber attack, and thus have the potential to detect both known and previously unknown forms of attack ([8] reviews such systems). For these, a range of tests run without the attack present can be used to assess the frequency of false positives ("false alarms") and their consequences.

When running tests with the attack present, the kinds of measurements depend on the kind of defense. For example, the effectiveness of a defense that is intended to *prevent* an attack would be measured in terms of how often it is able to do so, whereas the effectiveness of a defense that is intended to *respond* to an attack that has already begun would be measured in terms of not only how often it recognizes the attack, but also the speed of its recognition and the speed and adequacy of its response.

The work reported here is not focused on the specific challenges of anomaly-based intrusion detection, but on the broad range of pragmatic considerations of cyber defense testing. With this in mind, it is easy to see that factors of resource utilization (CPU load, bandwidth utilization, etc.) are potentially important. A defense that only functions well in benign resource usage conditions would likely be deemed to offer insufficient protection unless those benign conditions could be expected to be present in the operational system (and even then we might worry that a combination of attacks could violate this guarantee). Likewise, a cyber defense that, while effective at performing its defensive function, overly interfered with the conduct of normal work, would also be deemed unacceptable.

This "Defend" step focuses on collecting these measurements while operating in the test environment. If the defense is both acceptable when the attack is not present, and shows sufficient capability to thwart the attack, then the next step in the workflow, "Verify," will address extrapolation of the measurements to the operational environment and gauging their acceptability.

4.4 Verify

The purpose of this step is to verify the defense with representative(s) of the system user community to determine whether the defense is acceptable as is, or needs improvement.

This step relies upon a combination of extrapolation and estimation. The results from the tests run in the simplified scope and size of the test environment require extrapolation to what might be expected to occur in the real (operational) environment. The acceptability of those extrapolated results has to then be estimated. A performance model, should one exist, would be helpful in predicting performance metrics such as the defense's consumption of computational resources when it is scaled up to the real environment (e.g., the computation costs of monitoring the dozens of real users instead of the single user modeled during testing). The representatives of the user community are needed to estimate whether a defense will be acceptable. For example, if a defense places what seems like a small additional burden on the user by requiring an additional log in step, its acceptability will depend on how often that step has to be performed. An extra log in step per entire user session might well be acceptable; an extra log in step every time any piece of software is started less so.

This step involves the successful integration of inputs from two different types of stakeholders—the cyber defense experts and the user community. The cyber defense experts will be well versed in comprehending the details of their testing, but relatively unfamiliar with the operational environment and the needs and preferences of its user community. Conversely, the user community will have an intuitive understanding of their work practices and how to differentiate between acceptable and unacceptable changes to those practices, but may be challenged to infer the extent of those changes from the cyber test results. To be successful, this step requires effective communication between these two groups.

If as a result of this step the defense is deemed acceptable by the representatives of the user community, then progression to validation follows. If not, then the workflow will return to the Defend step to improve the defense, informed by the shortcomings identified from this verification step.

A complication could arise when multiple defenses are thought to be needed—when assessed individually they may seem acceptable, but in total may comprise an excessive burden (with respect to computation and/or usability). Ideally they would be treated as a portfolio, with selections from that portfolio evaluated together. This might not be feasible, especially should new vulnerabilities be uncovered in the future that warrant extension of defenses. The systems engineering practice of setting aside margin could apply here. In this case, the margin would be of computational resources set aside to accommodate future defense needs. When defense is proposed, its evaluation would balance how much of that margin it would consume against its criticality.

4.5 Validate

The purpose of this step is to validate the defense by running it (and perhaps the attack), in the real environment and comparing the results with test predictions.

Conducting this validation run must be carefully planned and coordinated with the users of the real system. When the validation run includes running the attack, it must be tolerable for the attack to succeed (presumably because the defense failed to work as intended). For attacks that threaten to corrupt or destroy data (including programs of course), it is essential to backup the system state in advance, and be able to restore it should the defense prove unable to prevent the attack in this validation setting. For attacks that threaten to exfiltrate sensitive data, the success of the attack must not lead to sensitive data being exposed externally; this could be prevented either by performing the validation run on "dummy" data (not actually critical, but handled as if it were), or by ensuring the exflitrated data is in fact contained within an outer protected scope. For attacks that threaten disruption (e.g., denial of service attacks), it must be tolerable that such disruption might occur during the validation run (e.g., by performing the validation run during a non-critical period of operation).

Whether or not an attack is included in the validation run, in all cases the possibility of the defense itself proving to be disruptive (e.g., inhibiting user access, killing valid processes, slowing or crashing systems) must be tolerable for the duration of the run.

It is important that the validation testing covers a comprehensive set of conditions. For example, akin to traditional stress testing, expose the defense to the conditions predicted to be the most challenging (e.g., maximum load from user program execution so as to determine how fast the defense will react under those circumstances).

If the defense proves acceptable, then transition to the "Deploy" step. If not, first determine why the verification did run not correctly predict the outcome of validation. Once this is understood, then it becomes possible to consider whether the defense may be improved and re-verification attempted.

Unanticipated discrepancies between validation run results and those predicted following the Verification step need to be examined to determine their root causes. These could have originated in any of the workflow steps prior to Validate, and the workflow would have to revert the earliest such step (hence the workflow figure shows an arrow going back as far as the Set Up stem). In more detail:

> *During the set up step* misunderstandings of the system and usage scenario to be defended, or misunderstanding of the attack (leading to mis-design of the defense), could have been the cause. Potentially the most challenging to avoid in advance is mis-configuration of the test environment by having omitted and/or simplified nuances of the operational environment that turn out to be important.
> *During the attack step* measurements taken of the (undefended against) attack may have failed to encompass all its detrimental effects, with additional ones discovered during validation. Thus, even if the defense succeeded in nullifying the measured effects, its failure to adequately nullify the previously unmeasured ones may diminish the appeal of the defense.
> *During the defend step* failure to sufficiently cover the range of conditions in which to conduct tests could be the cause. For example, this kind of failure could have led to under prediction of the false positives (false alarms) rate for an intrusion detection defense.
> *During the verify step* misunderstandings between the two groups involved in this step (the cyber defense experts, and the user community) could be a cause; another cause would be incorrect extrapolation from test results to operational. For example, discrepancy of a prediction formed by extrapolating response times measured during validation would suggest flaws in the extrapolation step. Note that discrepancies in either direction—under- or overprediction—are both of interest. For example, underprediction of how long it would take a defense to respond indicates that the verification step is at risk of advancing inappropriate defenses into the Validation step, wasting scarce time and effort in the real environment. This would be akin to advancing an immature software system too soon from unit testing to integration testing, resulting in wasting precious integration time on debugging errors that more effectively would have been caught by continued unit testing. Conversely, overprediction of how long it would take a defense to respond indicates that the verification step is at risk of inappropriately dismissing defenses that would actually be appropriate, not only wasting the time and effort it took to develop those defenses, but also potentially leading to deployment of sub-optimal defenses when better could have been done.

4.6 Deploy

The final step, assuming successful progression through the workflow to this point, is deployment of the defense to become a "normal" part of the real environment.

This might best be done first in a "probationary" period, during which the effects of the defense are carefully monitored, and the results of that monitoring are diligently scrutinized to catch early indications of incipient problems, or, hopefully, increase confidence in its acceptability for continued operation.

Even beyond such a "probationary" period continued monitoring of the deployed defense is highly desirable to

be able to recognize if/when the real environment evolves in ways that could undermine the acceptability of the defense (e.g., from a change of usage patterns of the user community).

5 Illustration of the workflow: a reconnaissance attack and a defense to it

This section reports on our experience following the workflow through several of its steps.

Our studies are based on an example of a "reconnaissance attack" that our colleagues have used to elucidate issues of cybersecurity in the JPL environment [1]. Our colleagues had already shown means by which the attack could be detected. For purposes of experimentation with our V&V workflow, we extended their defense to include an automated response that terminates the ongoing attack. We begin with an initially somewhat naïve defense, and show how the workflow (specifically the Verify step) leads to discovery of its unacceptability, motivating consideration of subsequent improvements.

5.1 Set up

We introduce our setting and the reconnaissance attack as we describe the set up step.

(a) *Identify the system and scenario to be defended*. Our setting is the Jet Propulsion Laboratory. We focus on the combination of personnel and computing capabilities responsible for commanding a space mission. There is the need to protect the confidentiality and integrity of the information within this environment. This constitutes the "system" we wish to defend. Note: while this is obviously a setting particular to space agencies, the same considerations would apply in many terrestrial settings (e.g., control of an industrial plant). The scenario we consider is the daily activity of a legitimate user within that system, who logs on, performs his or her work duties (e.g., helps plans the commands to be sent to direct the spacecraft), etc.

(b) *Design or select the cyber attack to be defended against.* As already mentioned, our studies are based on the "reconnaissance attack" presented in [1]. Its starting point assumes an initial breach during which an adversary has gained brief write access to a legitimate user's home directory (there are multiple plausible ways in which such a breach could occur). The adversary exploits this breach to add to the user's login script a few lines that will automatically open up remote display terminals on the attacker's machine whenever the legitimate user logs in. These lines start a background xterm process set to display on the attacker's machine, thus allowing the attacker to execute as the user (a capability that persists even after the user has logged out), and invoke the xkibitz command also set to display on the attacker's machine, thus allowing the attacker to view all the user's subsequent activity (http://www.linuxcommand.org/man_pages/xkibitz1.html) [1] go on to explore the further ramifications of this attack (how it may continue in a plausible scenario of user activities), which for simplicity here we do not address.

(c) *Determine that the cyber attack works—through analysis or (if acceptable) testing in the real environment.* In this case, it is safe to demonstrate the viability of this attack in the real environment, since the attack is something we may conduct ourselves, routing the remote terminals to one of our staff posing as the attacker for demonstration purposes. As long as our attacker is careful to only observe and issue a benign command to demonstrate ability to control, no harm is done. In contrast, an attack that disrupted normal work (e.g., an attack on availability to key resources) or made changes to project artifacts (an attack on integrity) would not be safe to experiment with in our environment without advance preparation.

(d) *Configure the test environment to model the real environment as required for testing.* Our colleagues have been developing a "Cyber Defense Research Laboratory", isolatable from the institutional network, in which to safely perform cybersecurity experiments. In the virtual environment the laboratory provides it is possible to mimic all the physical devices needed to simulate the real environment. However, the limited computational resources of the virtual environment may mean that its simulations execute more slowly than reality. In that case, the laboratory could be augmented with additional hardware resources, if the cost of doing so justifies making the lab more closely match the real environment. The laboratory could, of course, be augmented with additional hardware resources if the cost of doing so was thought justified to more closely match the real environment. Virtualization makes it possible to achieve an adequate level of fidelity as efficiently as possible. This enables the rapid assembly, configuration and cleaning up of experiments in the laboratory. Using these capabilities, they set up a test environment to recreate a "close approximation of a real-world mission architecture" in which to demonstrate the "reconnaissance attack". Their set up comprised computational resources representative of (but fewer in number than) the actual operational environment, relevant infrastructure (in particular, a network), and an attacker's computer with connectivity to that network. Furthermore, they took a step further: they made available to us a virtualization of the environment—one that recreated just the essence of the reconnaissance attack—small enough to run on a standalone PC or Macintosh®.

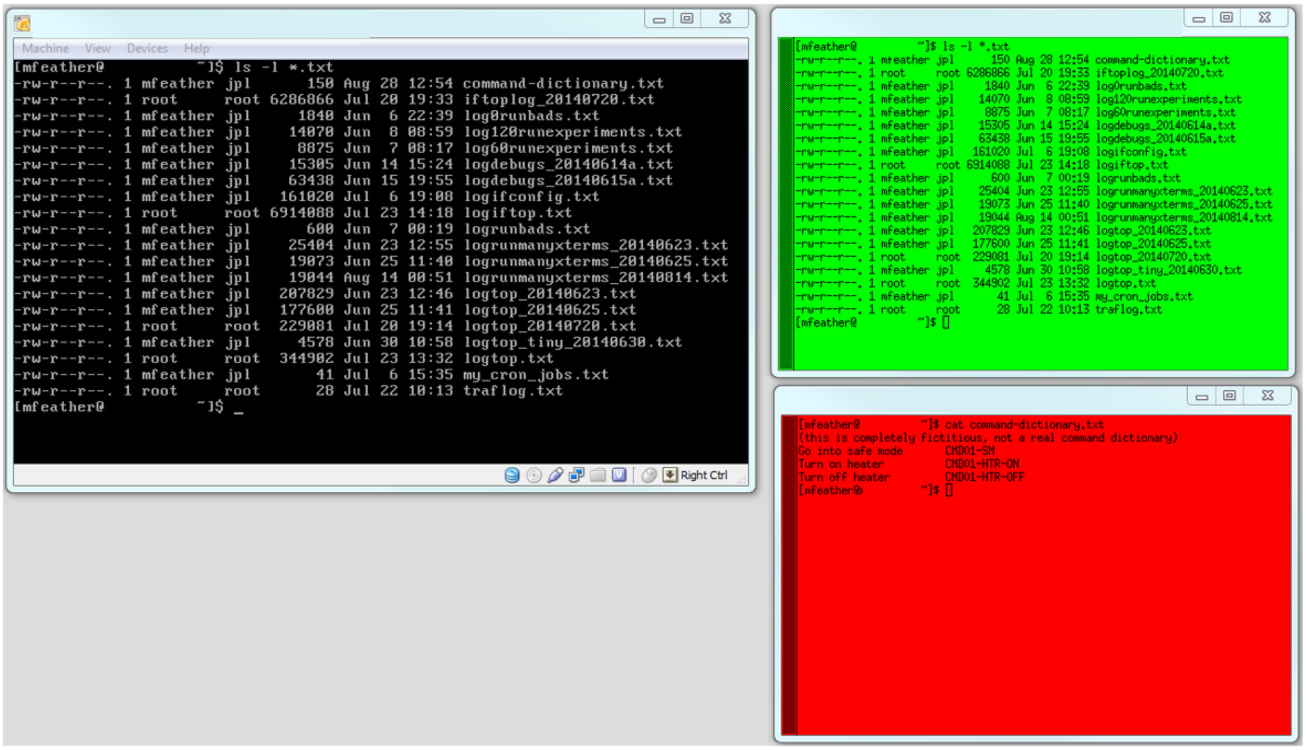


Fig. 1 Screenshot of running the attack in a virtual environment (machine names obscured)

This proved very convenient for us to experiment with the attack and defenses to it. The results reported in this paper are from running in that small virtualization. We have subsequently transferred out defenses back to the laboratory and confirmed that they run as expected there too.

5.2 Attack

This step is to determine whether the attack succeeds *in the test environment* without the defense present and active, and gather measurements of the attack.

Our colleagues had already demonstrated the full extent of the "reconnaissance attack" in the test environment. It is easy to see that it succeeds—when the legitimate user logs in, the anticipated remote terminals appear on the attacker's machine, and do in fact give the attacker observation of the legitimate user's actions, and the ability to issue commands as that user. Furthermore, the ability to issue commands persists even after the legitimate user has logged out (as long as the machine(s) the user logged into are still running). Figure 1 shows the attack running in the virtualized environment running on a PC. The black window is the victim's window; the victim has just issued a command to list files. The green window is the attacker's view, via the xkibitz process, of the victim's window. Visible in this window is the victim's command and the directory listing that resulted. The red window is the attacker's xterm by which the attacker can execute commands any user commands the victim could. It shows the attacker listing the contents of one of the victim's files (note: the file seen is named "command-dictionary.txt" to suggest exfiltration of sensitive information, but its contents are completely fictitious for presentation here).

We characterize this attack as threatening both confidentiality and integrity. Measurement is a straightforward assessment—confidentiality of the user's activities is totally breached, and the extent of the threat to integrity spans whatever the user can do without requiring re-authentication.

5.3 Defend

This step is to design and implement a proposed defense and test it within the test environment.

Our colleagues had already demonstrated one form of defense—they configured a commercial network monitoring system to recognize symptoms of the "reconnaissance attack" (one symptom being a process running on the machine the user had logged into but displaying elsewhere), and showed that the monitoring system would detect the symptom; they coupled this with a response, to alert a defender—another legitimate user on the network, charged with overseeing the monitoring system's alerts and responding accordingly.

The defense we consider here is a modification to this. It replaces the response to alert a human defender with *automation* to respond to the alert by killing the attacker's

processes (those detected by the aforementioned monitoring) and by logging the activity for deeper forensics later (e.g., to help to determine how the breach occurred in the first place). We implemented this as a script that kills the attacker's processes on the user's machine, this script to be invoked by the network monitoring system. However, in the environment our colleagues had set up, the network monitoring system has a distributed architecture. A data-gathering component is running on each machine, forwarding data to a central machine that runs the code to determine whether the data exhibit any of the symptoms being monitored for, and if so invokes the corresponding script. The reason for this architecture is that if it was to be put to widespread use, it would keep the symptom monitoring computations off the users' machines to minimize the additional computational burden on those machines. Given this architecture, our script, being run on the central machine, has to have the appropriate authority to kill processes on the user's machine. In our automated defense, we achieve this by giving the machine that executes the script the authority to run passwordless-ssh as the user on the user's machine (where the attackers processes are executing) so as to be able to kill those processes (we created a pair of authentication keys for this purpose).

We wrote, tested and debugged the script in the test environment until it was running smoothly when invoked by the monitoring software. We were then ready to take measurements:

First, we then took measurements of overall resource usage when the attack was *not* present (i.e., without malicious code added to the user's login script). Even though there was no attack during these tests, the monitoring software was active, and the purpose of the tests was to measure the computational load it put on the system—very little as it turned out.

We then repeated the tests, but with the malicious code present in the user's login script (and the defense active). We observed that the action of logging in by the legitimate user opened the remote terminals on the attacker's machine, but shortly thereafter the processes running those were killed and the events logged. We manually measured the time it takes between those remote terminals opening and the defense acting to close them. These durations varied from a few seconds to close to a minute. We recognized that at least some of the variation was due to the periodicity of the monitoring software—e.g., if an attack was started shortly before the next monitoring period, then it would soon be detected. This variability led us to conclude we would be wise to run a large number of tests to determine the distribution of these durations. To make multiple tests easy to perform, we simplified the attack to its essence—the opening of an xterm from the user's machine to a remote machine. We could cause this to happen repeatedly (and automatically), without needing to have the simulated user log in each time. We also automated the gathering of the measurements—CPU load, memory utilization and network utilization.

Finally, we repeated the automated tests yet again but in different conditions. We were specifically interested in what would happen if:

(a) the user's machine was subject to different processing loads (speculating that this might slow the portions of the cyber defense monitoring and/or response running on the user's machine), and
(b) the user's machine was involved in sending or receiving network traffic (speculating that this might interfere with the communications needed for the cyber defense monitoring and/or response).

Having gathered these measurements, we were ready to attempt to extrapolate the test environment results to the operational environment, and to assess the proposed cyber defense's acceptability (in terms of costs, risks, and benefits) to the user community.

5.4 Verify

The purpose of this step is to verify the defense with representative(s) of the system user community to determine whether the defense is acceptable as is, or needs improvement.

First, we set about comprehending the data we had gathered. Poring over the raw log data was unsatisfactory for this purpose, as can be appreciated from the snapshot of portions of these log files (Fig. 2).

Following the precepts of information visualization, we began development of a mobile dashboard to be used for communicating and understanding these kinds of metrics. Indeed, application of information visualization to cybersecurity data and its analysis is widely advocated, e.g., [7].

Figure 3 shows a visualization of CPU loads on the victim's machine. Total CPU load is shown by the upper, orange-colored, line; the portion of this consumed by the

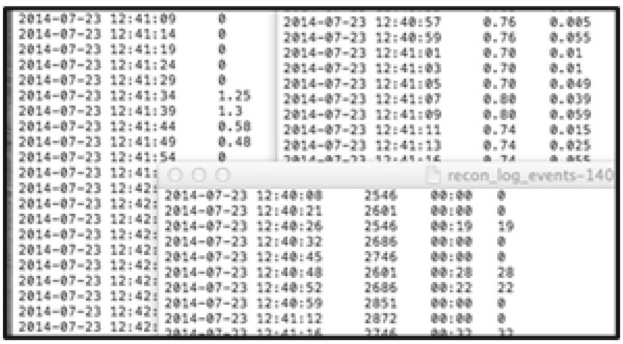


Fig. 2 Log files

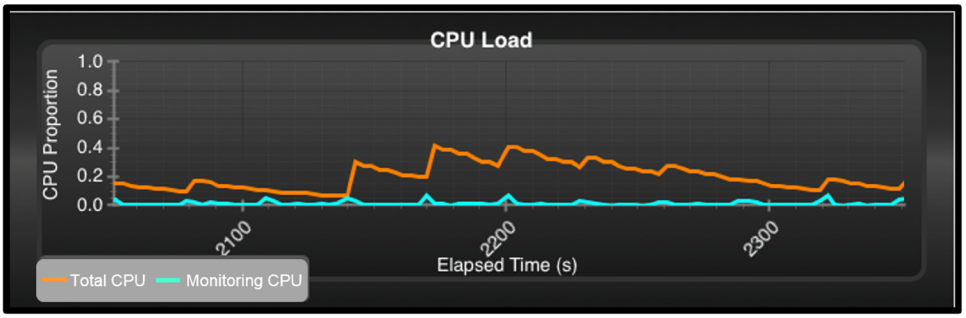


Fig. 3 Visualization of CPU loads

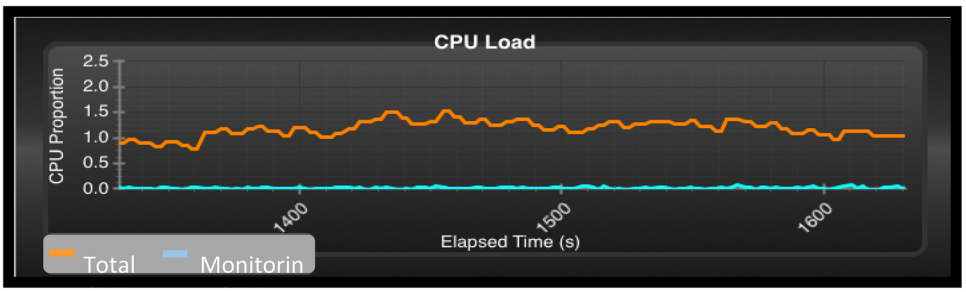


Fig. 4 Visualization of CPU loads under heavy conditions

monitoring system running on the victim's machine (gathering and forwarding data to a separate machine to analyze the data) is shown by the lower, teal-colored, line. This clearly shows that the monitoring system consumed only a small fraction of the CPU within our test environment.

Results of another test run, conducted while the additional processing was added to deliberately load the CPU with enough tasks to keep it fully occupied (and then some) most of the time, even more dramatically showed the relatively inconsequential load of the monitoring system—Fig. 4. (note that the vertical scale has expanded relative to that of the previous figure).

Since these were measurements taken in the test environment, we had to consider what might happen in the operating environment, where there are dozens of machines and users all of whom would be candidates for this monitoring-based defense. To consider this, we engaged in discussions with the person responsible for cybersecurity on one of the current flight projects. We determined that if the monitoring system were to be deployed, it would be architected to have a dedicated machine to receive the monitored inputs forwarded from all the users' machines, perform the analyses to detect symptoms of cyber attacks, and trigger responses as appropriate. On this basis, we felt confident that our measures in the test environment showing the insignificant CPU load from the forwarding portion of the monitoring system on the user machine indicated the same would hold of user machines in the operating environment. We also took measurements in the test environment of the memory used by the forwarding portion of the monitoring system on the user's machine. These showed it to be a small, static amount (and did not need visualization to understand). Again, we concluded this would hold true in the operational environment.

We also took measures of network traffic and generated similar visualizations of these data (which showed the network traffic from the monitoring to be inconsequential).

We next considered the effectiveness of the defense, measured by the durations over which the attacker xterm was open (the shorter the better, of course). In our initial test runs, we had allowed sufficient time between successive "attacks", i.e., opening of remote xterms, for the periodic monitoring-based defense to have time to detect the remote xterm and trigger the response to have its process killed. We timed each attack to start randomly in a uniform interval of 30 s (the monitoring period) to avoid accidentally synchronizing with the periodic monitoring. This allowed us to gather hundreds of simulated attacks. However, this was quite a time consuming experiment—allowing at least 60 s between successive attacks, and adding on between zero and 30 s more, meant a run of several hundred attacks would last many hours. Impatient with this, we made one further adjustment to our experimentation, to start successive attacks without waiting sufficiently long for the previous one(s) to have been detected and responded to. This allowed us to run hundreds of attacks in an hour or so. The visualization we developed to view the results is seen in Fig. 5. Each horizontal line segment denotes the duration of the attacker's xterm process (the varying colors are used to visually distinguish different segments, and otherwise do not have a specific meaning). It can readily be seen that their durations vary from a few seconds to a few tens of seconds. A histogram from one of our runs of several hundred attacks shows this to be so—Fig. 6.

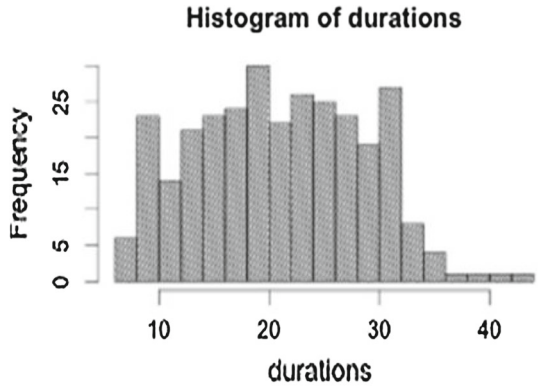


Fig. 5 Histogram of attack durations

We arranged the combination of our visualizations and statistics as an iPad® application—Fig. 7. This takes as input data from a test run, additional images if any (e.g., the histogram), and generates the summary statistics and visualizations. The latter are coordinated with a single slider that allows the viewer to scroll back and forth through an entire test run's results.

Having generated this cogent presentation of test results, we were ready to discuss with the security expert of one of the operational flight projects extrapolating these results to an operational environment and assessing the benefits, costs and risks of our prospective defense.

Benefits From the defense times measured in the test environment (the number of seconds it takes between initiation of the attack, and the automated defense terminating the attacker's processes), we estimated that similar times would apply in the real environment (assuming the same periodicity of monitoring). As we did this, we considered factors that could potentially make a significant difference in timing. For example, in the real environment the need to monitor dozens, possibly hundreds, of legitimate users and their processes might slow down the detection time. We postulated that in the real environment adequate computational resources (on machine(s) dedicated for this purpose) would be devoted to analyzing the information from monitoring to detect the symptoms of an attack, and that, therefore, there would not be a dramatic slowdown compared to the times measured in the test environment. On this basis, the responsiveness of the defense was deemed to be adequate. For this particular defense, no additional benefits were identified.

Costs CPU and memory loads on the user's machine were measured in the test environment. These stemmed from the need to monitor that machine's interaction with the network (specifically, to monitor for outgoing xterms) and forward the information to an analysis capability on a separate machine. In the test environment, the measures showed these loads to be relatively small, and our expectation was that they would be no worse on users' machines in the real environment. On this basis, these resource costs were deemed to be acceptable. In terms of user inconvenience, since the defense is automatic, no additional actions are required to be performed by the legitimate user.

The expectation was that a network monitoring system would be present for other purposes, so our defense might be asked to contribute to its costs, but would not have to pay for the whole thing.

One huge drawback to our naïve defense was that it precludes the user from opening *any* xterm to a remote display. This curtailment of a highly useful capability was deemed unacceptable, indicating we would need to rethink our initially proposed defense.

Risks In addition to the drawback just mentioned, consideration of the defense's use of passwordless-ssh as the means to give the defense script (executed by the central machine), the authority to kill the suspicious processes on the user's machine was also deemed a significant drawback. In cybersecurity terms, it violates the principle of least privilege—passwordless-ssh grants unnecessarily much authority relative to the purpose it is achieving. This too was an indication we would need to rethink our initially proposed defense.

Overall status The overall status of our defense is portrayed in the radar chart seen in Fig. 8. Of its eight key factors,

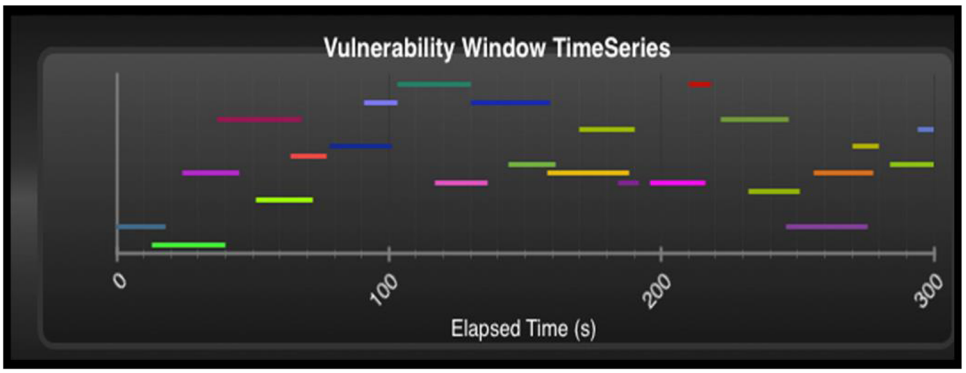


Fig. 6 Visualization of attack durations

Fig. 7 Dashboard for display of metrics and visualizations

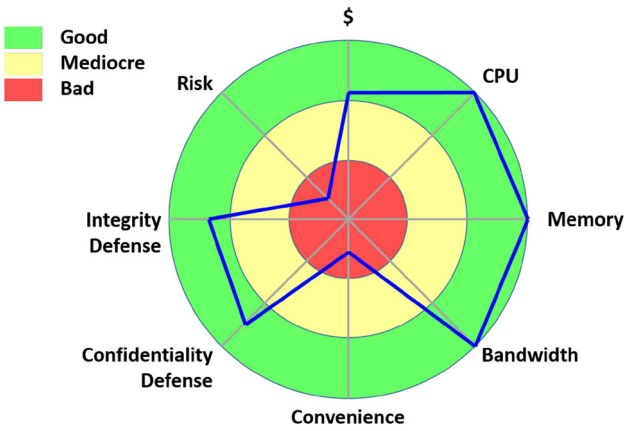


Fig. 8 Status of our initially proposed defense

six of them fall into the "Good" annulus (meaning they are acceptable); however, two of them (risk and convenience) fall into the "Bad" center (meaning they are unacceptable).

5.5 Return to the defend step to revise the defense

The recognition of flaws in our initially proposed defense renders it unacceptable, so we do not advance to Validate. Instead we return to the workflow's defend step to attempt to revise the defense accordingly. We briefly summarize our efforts and thinking in this direction:

To allow legitimate users the capability to open remote xterms, our planned revision of the defense maintains a "whitelist" of user-approved external locations to which remote xterms are to be allowed to be displayed. The user is provided a mechanism by which to add destinations (IP addresses) to this whitelist. The defense script, when triggered, looks to see whether the remote destination of the xterm of the suspicious process is included in the current whitelist, and only if it is not included does it go ahead and kill the suspicious process. We have implemented this, and expect it will not unduly add computational burden, but have yet to perform the tests to verify that this is so. However, it still might prove unacceptable to the user community. For

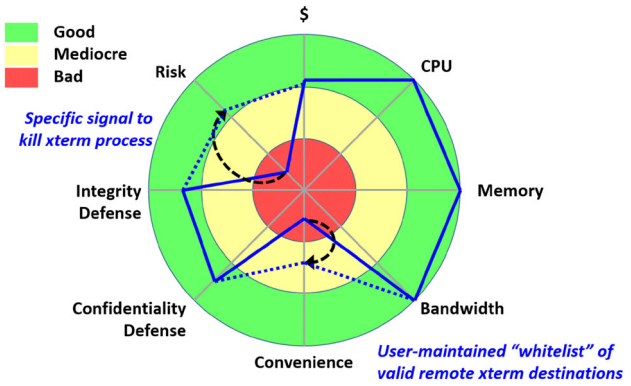


Fig. 9 Radar chart of revised defense

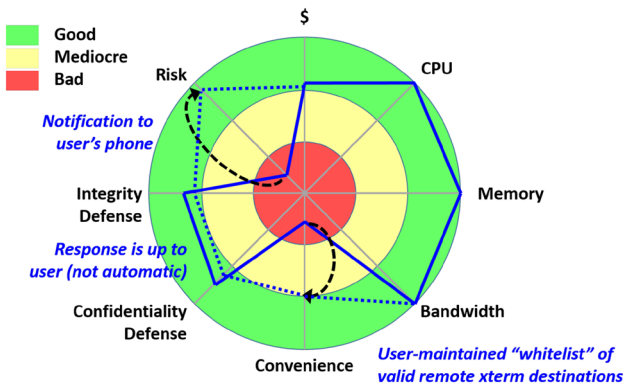


Fig. 10 Radar chart of another defense

example, there is the risk that during critical phases of mission operation the need to whitelist additional IP addresses could introduce a critical delay. Users may have long since populated their whitelist with the regularly used IP addresses, but now an unusual circumstance requires them to communicate information to an atypical destination. Especially under time pressure, we would not want them to have to struggle to recall how that whitelisting was performed.

To reduce the unnecessary risk that passwordless-ssh entails, the response to detection of an unapproved outgoing xterm needs to be given the specific authority to trigger the killing of that process, but no broader authority. We have yet to implement this.

We have not implemented this change yet. However, once we do, our assessment of the defense would change as shown in Fig. 9. With a specific signal to kill the attacker's xterm process, the risk might be gauged to have become acceptable (there is still some risk, e.g., if an attacker could find a way to spoof that signal). The inconvenience of not being able to legitimately open *any* remote xterms has been resolved, but there remains the inconvenience of users having to explicitly whitelist allowable xterm destinations. Whether this is deemed acceptable or not is something the intended user community would have to gauge.

We have also *postulated* an alternate revision, in which the response does not automatically kill any process, but simply notifies the legitimate user in question (e.g., by sending a notification to the user's phone). This could still be coupled with a "whitelist" feature so that the user could suppress notifications of outgoing xterms to those whitelisted locations. The status of such a defense might change from out naïve initial proposal as shown in Fig. 10. The risk from inadvertently killing legitimate processes has disappeared, and the level of user inconvenience has improved to acceptable (perhaps). However, merely notifying the user in question is weaker than automatically killing the attacker's processes—the user might not respond to the notification immediately, for example. This is indicated by a slight decrease in the Integrity Defense and Confidentiality Defense factors.

6 Conclusions

We have addressed the decision of whether to introduce a cyber defense into an operational environment. The decision will hinge on the key metrics of the costs, benefits, and risks of the cyber defense. We have argued that to measure these it will be desirable to first test the cyber defense in a safe, isolated environment. However, this raises the challenge of fidelity between the test environment and the operational environment. To address this, we have described a workflow modeled after systems V&V testing, with the steps of Set-Up, Attack, Defense, Verify, Validate, and Deploy. We have discussed the collection of metrics in the test environment, and their extrapolation to the operational environment.

Our workflow is illustrated with an example of a reference attack and defenses to it. We show our use of appropriate visualizations to cogently present the gathered metrics, and summarize the status of the several possible defenses.

Our focus has been on assessing individual defenses. In making the decision of which defenses to introduce, it will also be necessary to assess the risk posture of the overall operational environment. An approach such as that described in [2] could be employed for this purpose.

Acknowledgments This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. We gratefully acknowledge the members of JPL's Cyber Defense Research Initiative for many fruitful discussions, use of their test environment and infrastructure, and use of their illustrative "reconnaissance attack."

References

1. Byrne DJ, Morgan D, Tan K, Johnson B, Dorros C (2014) Cyber defense of space-based assets: verifying and validating defensive

designs and implementations. In: Conference on systems engineering research (CSER 2014) procedia computer science, vol 28, pp 522–530
2. Buckshaw DL, Parnell GS, Unkenholz WL, Parks DL, Wallner JM, Saydjari OS (2005) Mission oriented risk and design analysis of critical information systems. Mil Oper Res V10:N2
3. Craven P, Ramachandran N, Vaughn J, Schneider T, Nehls M (2012) Test before you fly—high fidelity planetary environment simulation. In: Global space exploration conference (GLEX)
4. DeVale JP, Tan KMC (2010) Evaluating information assurance performance and the impact of data characteristics. In: 2010 IEEE international conference on technologies for homeland security (HST). IEEE, pp 15–21
5. Dumas LN, Walton AL (2000) Faster, better, cheaper: an institutional view. Acta Astonautica 47(2):607–621
6. FIPS PUB 199 (2004) Standards for security categorization of federal information and information systems. Computer Security Division, NIST. http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf
7. Jacobs J, Rudis B (2014) Data-driven security—analysis, visualization and dashboards. Wiley, New York, ISBN: 978-1-118-79372-5
8. Jyothsna V, Prasad VVR, Prasad KM (2011) A review of anomaly based intrusion detection systems. Int J Comput Appl 28(7):26–35
9. Mirkovic J, Benzel TV, Faber T, Braden R, Wroclawski JT, Schwab S (2010) The DETER Project: advancing the science of cyber security experimentation and test. In: Proceedings of the IEEE HST; 10 conference, Waltham
10. Wueest C (2014) Threats to virtual environments, Symantec. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/threats_to_virtual_environments.pdf (retrieved 1 Sep 2014)