AN ESTIMATION OF ATTACK SURFACE TO EVALUATE NETWORK (IN)SECURITY

Andrea Atzeni and Antonio Lioy
Politecnico di Torino, Dip. Automatica e Informatica, Torino, Italy

Keywords: Security metric, attacker surface, network security.

Abstract: In spite of their importance, security measurement methods are unusual in practice. Security assessment is left in the hands of personel security experts' judgment, with poor formal arguments on the security level of the underlying system. Thus, it is difficult to distinguish among security alternatives or justify possible security changes or improvements. In this work we focus on a limited but important set of security indicators, suitable to estimate the attack surface a system exposes, thus introducing a simple and objective metric for a fast evaluation of an important security facet.

1 INTRODUCTION

Given the importance of computer-assisted activities, protection of computer systems is of the utmost interest (Atzeni and Lioy, 2005), but *how to evaluate if a certain level of "protection" is enough*?

To make objective security choices, we need objective measuring tools, capable of assessing the cost of a protection technique, adequate to define formally the system security level, fitted for cost-benefit economical analysis. In just few words, there is the need to define an objective security metric.

The problem of defining a metric, both easy to use and widely adoptable, is far from simple. This is because security is a wide concept, including many different and often not well defined aspects. Even if many past approaches attempt to introduce measurement methods, there is presently a lack of solutions for practical use.

In this paper, we introduce a metric to evaluate the attack surface of a given system, this is to estimate the ways and the likelihood which an adversary can enter the system and potentially cause damage. In other words our purpose is to evaluate the attacker opportunities, since it is quite intuitive the existence of a proportional relation between the number of available attack opportunities and the probability of a successful attack (Howard et al., 2003).

2 RELATED WORK

Different approaches have been proposed to evaluate the security level of a computer system.

In governments or large organizations security of the final system is commensurate to the development process security, and/or quality of the management procedures, that is the maturity level of the organization (CC, 2006; SSECMM, 2003). Such approaches involve large human groups and time-consuming analysis, so they are not very well suited for systems of small companies.

Other evaluation proposals try to measure the system focusing on its usual operations. In vulnerability scanning techniques (Budiarto et al., 2004) a vulnerability scanner interrogates network hosts about port status (open vs. closed), the configuration of the operating system and available services. Such a test discovers known bugs, but does not find unknown weaknesses. In spite of such a limitation, it is probably the most adopted tool in security analysis, and in our work we assume the use of similar tools.

In red-team attack evaluation (Schudel and Wood, 2000) an "evil band" is created to break the system. The needed time and the cost of adopted tools are both suitable metrics. Such an approach suffers of the little repeatability of the experiment, due to the individual characteristics of people employed.

Model-based evaluations establish a model describing the behavior of the system and then foresee the possible security level. In this set we include *reliable block diagrams*, *fault trees* and *attack trees*, and *model checking* techiniques (Nicol et al., 2004).

Applicability to the security field is promising but not simple, due to the lack of a formal and validated security-behavior model.

Focusing on the field of attack surface definition, previous works mainly address the surface of a soft-

Atzeni A. and Lioy A. (2007).
AN ESTIMATION OF ATTACK SURFACE TO EVALUATE NETWORK (IN)SECURITY.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 493-497
DOI: 10.5220/0002377304930497

ware artifact (Howard et al., 2005), rather then the surface of a complete system. Thus, even if the underlying idea is the same, our method is very different due to the different scope of application.

3 APPROACH

Fundamental properties of a measurement system are ease of use and broad applicability, hence the need to assess security aspects both common to every host in a network and simple to be painlessly measured.

In our schema, we inspect different attack surfaces of a network considering different *attacker viewpoints*. Thus, we obtain different values reflecting different locations of the attack starting point. We think this analysis is especially helpful to determine where an attacker presence would be most dangerous and to which *user role* (e.g. business manager, clerk, Internet user)[1] the network exhibits the most dangerous weakness.

The proposed network metric tries to evaluate the likelihood of attacker opportunities by two different approaches. The first is based on an indirect quantification of risk, which assumes a relation between the quality of the software artifact implementing the network service and its security. The second approach evaluates network security on the base of known vulnerabilities. The two approaches are combined to compute the attack surface area of the networked system under analysis.

About the attack model, in this work we assume that the enemy is *outside*, i.e. someone different from any legitimate user, and so can attack a host with no exploitable privilege. The attacker is supposed to be able to reach internal hosts in the network, through network zones not protected by firewalls, or through firewalls which do not filter the ports. In such a scenario, network infrastructure (e.g. firewalls, switches, gateways, links) is considered perfect, in the sense that a filtered port is unreachable from the opposite part of a firewall, and network apparatus are invulnerable to attacks.

4 A SIMPLE ATTACK SURFACE MEASURE

An attribute characterizing every host in computer networks and simple to compute is the number of open ports in a host. An open port represents a possible entry for an enemy attacker, e.g. a possible link to a buggy application, which may lead to a possible unobstructed passage to system control. Hence, each open port represents a possible threat to the security of a host and conceivably of the entire network. Such a measure approaches the problem in a probabilistic way, not considering the service's peculiarities, but only its mere presence. So, we define a function $F(n)$[2] representing the security of a host, or, tantamount to our purpose, the insecurity, where n is the number of open ports. So, the function must be monotonically increasing with the number of ports, and defined for all $n \in [0, 2^{16}]$

Based on the previous observation, the simplest *probabilistic service* insecurity level of the system is $\mathcal{S}(n) = \frac{n}{65536}$, that is the number of open ports, each with an insecurity value of $\frac{1}{65536}$, and with the maximum insecurity value 1, when all the ports are open. In this sense, the value 1 is a very undesirable one, which should be never achieved in any real-life network.

The choice of actual function needs further research, supported by real experiments, but the above example, in spite of its simplicity, shows desirable properties of (inverse) proportionality among the value and host security.

For the entire network, a straightforward extension involves the computation of open ports for each single host, thus the security function depends on two variables, $F(n,h)$, where h represent the hosts inside the network. A simple suitable example formula is $\mathcal{S}(n,h) = \sum_{i=1}^{h} \frac{n_i}{65536}$, where h represents the total number of hosts in the network. To pay attention to attacker position we define the *subjective attack surface*, that is the attack surface considering the viewpoint of an attacker operating from a specific starting point. This is evaluated by considering the capabilities of filtering devices.

This can be expressed by a function $F(n,h,a)$ where the new parameter considers the presence of firewalls in the evaluation formula, able to block the ports b_i, like, for example:

$$\mathcal{S}(n,h,a) = \sum_{i=1}^{h} \frac{n_i^a}{65536} \quad (1)$$

Where n_i^a represents the number of ports open and not filtered on the host i, considering the attacker viewpoint a. The value n_i^a is derivable from logics *and* operation between n_i and b_a, where b_a are the filtered ports between the attacker point a and the host i.

[1] supposing different roles access the network from different entry points

[2] for sake of clarity, we will indicate in the rest of the paper $F(.)$ as abstract function, and $\mathcal{S}(.)$ as real instance of the measuring function

Insofar, we discussed the services as abstract all-equal entities. Concretely, any service has its distinctive bugs and security flaws. If we know weaknesses of the service, we can apply the *actual attack surface level* as discussed in the next section, but even if there are not apparent security weaknesses, we may assume a correlation between the security of the service and the *quality* of the software artifact, and then weight the open port on the base of the service's quality

We can borrow from software engineering many evaluation systems, like LoC (Lines of Codes), suitable particularly in case of homogeneous conditions (Disney and Johnson, 1998), more abstract measurements like the Albrecht's Function Point (Albrecht and Gaffney, 1983), class-design metrics (Chidamber and Kemerer, 1994), cyclomatic complexity (McCabe, 1976), and others.

Since security is our primary concern, another possibility is to exploit the known vulnerabilities as a software quality indicator. In this case, software quality increases as the number of known vulnerabilities decreases. We may compute the total number of vulnerability for a specific product, the time interval in which the vulnerabilities manifested (e.g. the interval between the first product availability on the market and the present time), and thus the vulnerability frequency occurrence. From the set of all vulnerability frequency $\mathcal{F}_v$, we may determine the maximum and the minimum value, and we may define a function mapping univocally and proportionally elements of $\mathcal{F}_v$ to the interval $(0,1]$, and adopt the function result as quality coefficient q_j

The abstract formula becomes $F(n,h,a,s)$, where s contains information suitable to measure the quality of services, based on a specific software metric. Following with our example, we transform the previous example function in:

$$\mathcal{S}(n,h,a,s) = \sum_{i=1}^{h} \frac{\sum_{j=1}^{n_i} r_j^a \cdot q_j}{65536} \qquad (2)$$

Where r_j^a is a Boolean 1 or 0 value, respectively if the service is reachable from attack point a, and q_j is the quality coefficient of the software artifact realizing the service. Since the relation between software quality and software security is at the moment not well understood, the best quality metric to adopt will be object of further research.

Another aspect to consider is the level of interaction with the external environment, in particular if the service is of *read-only* type, not permitting change operation to external users and exposing internal data without elaboration of any kind (e.g. a yellow pages service) or rather the service is to some extent *read-write*, i.e. able to change data in response to user actions, constituting a greater danger for the system. Considering this aspect, the above formula becomes:

$$\mathcal{S}(n,h,a,s) = \sum_{i=1}^{h} \frac{\sum_{j=1}^{n_i} r_j^a \cdot int_j \cdot q_j}{65536} \qquad (3)$$

Where int_j is a corrective coefficient. We can simplify the practical application, without loss of generality, supposing int_j always equal to 1 in case of *read-write* service, and then by choosing the adequate (smaller) int_j coefficient in case of *read-only* services.

The actual coefficient depends on the organizational policies, in particular which amongst confidentiality, integrity or availability is the main security concern[3].

5 VULNERABILITY-DRIVEN ATTACK SURFACE

Another aspect of security evaluation we consider pertains to the actual and known weaknesses inside the network, and not their likelihood as in previous section. The overall evaluation formula now comprises the weaknesses in the system, as the variable v, assuming a form like $F(n,h,a,s,v)$. The evaluation process is slightly more complex and composed by a few steps, depicted in the following paragraph.

5.1 Weakness Classification

Ideally, an existing classification should be available in an ordered set, in which to every weakness is assigned a weight. Works exist this direction: dictionaries of vulnerabilities (MITRE, 2001) permit unambiguous identification of a vulnerability, vulnerability databases (Martin et al., 2002; NIST, 2005) store and classify known security problems, and regarding vulnerability's severity, many scoring systems come forth, even if they are suitable mainly in a specific development context (US-CERT, 2003; Microsoft, 2003; SANS, 2003).

Amongst the scoring systems, the Common Vulnerability Scoring System (CVSS) is in our opinion the most interesting (FIRST, 2005). This standard faces the problem of classifying the vulnerabilities in a systematic and computable way. It is the most formal and its rating is well comparable with our approach on hypothesized security. With the CVSS

[3]For example, if confidentiality is the primary concern even a read-only service may be source of great harm, thus the corrective coefficient in case of read-only and in case of read-write should be very similar

scoring system, in the example proposed so far, a single vulnerability in a service is approximately from one to two orders of magnitude greater than an open but not known as vulnerable service.

5.2 Network Analysis

Until now, we have supposed to link our insecurity to the number of open ports. In this sense, it is quite easy to perform the required network analysis. Services and running operating systems are identifiable by information gathering techniques, such as *OS fingerprinting*, banners, and trivial analysis of the service behaviour. Good tools to perform network-based security analysis already exist (fyodor@Insecure.org, 1998; Hauser and Revmoon, 2006). However, counting the number of open ports alone is a somehow simplistic approach, since some services use non-statically-defined ports, some computers do not listen to open ports (e.g. some P2P systems), and so forth.

Thus we face the problem to switch from a port-centric to a service-centric analysis, considering the services accessible from the "outside", not just looking at the open ports. To perform this task we could rely on a more sophisticated source of information, such as a formal description of the target system.

For example, the Positif and Deserec european projects have defined and use an XML dialect, named PSDL (Positif System Description Language), useful to describe a computer system. Moreover, these projects provide tools to describe security policies too, such as firewall filtering rules. Consequently, given a network description in PSDL and the related policies, which services are reachable from a network point is computable.

5.3 Result Evaluation

From the previous two steps the information needed for an evaluating function should be distillable. In particular, the network topology and services, the path between any two points, and the vulnerability coefficient for known vulnerabilities are available. As said, the general function is of the form $F(n,h,a,s,v)$ and v is a parameter related to known vulnerabilities. Assuming that a service attack surface value is composed both by its quality (that estimates the probability of unknown vulnerabilities) and by the known vulnerabilities, and that the two aspects are independent, we obtain as example of the evaluation function:

$$\mathcal{S}(n,h,a,s,v) = \sum_{i=1}^{h} \frac{(\sum_{j=1}^{n_i} r_j^a \cdot (int_j \cdot q_j + \sum_{k=1}^{v_j} vul_k)}{65536} \quad (4)$$

In this example vul_k is the single vulnerability coefficient, as evaluated by the CVSS system. Instead, v_j is the number of vulnerabilities for the service j of the host i. $\sum_{k=1}^{v_j} vul_k$ is equal to zero if no vulnerabilities are known for service j.

5.4 Threshold Evaluation

When a result is evaluated, a threshold $T(.)$ establishes whether the attack surface area is acceptable. The threshold should be evaluated based on the properties of the system, in particular its importance inside the organization, and the security policies (e.g. "the system must achieve a very high security level"). Such aspects are related, but of course they keep a degree of autonomy. For example, for a very sensitive host, an optimistic agency could leave a service available even if a vulnerability (not exploitable at the moment) is known, maybe in order to ensure performance. Instead, another, more pessimistic, agency may close the service and so increase the security level.

The threshold is then $T(i,p)$, function of system importance (i) and security policy (p). An example function may be very simple as in (5). For simplicity, the level refers to a single host, and if many hosts are present in the system under consideration, we suggest a multiplication for the coefficient h, that is the number of available hosts.

$$t(i,p) = i \cdot p \quad (5)$$

To enhance the clarity of the measure, we suggest the discretization of the value in a scale of few natural values, for example from 1 to 10 (where 1 means greatest importance).

The security policy, in the sense of the desired security level, may be a correction coefficient, that proportionally increases or decreases the importance i. For example, qualitative statements as low, medium and high (security level) can be respectively mapped onto the numbers 3, 2 and 1.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have proposed an evaluation schema for a simple estimation of the security level in a computer network, suitable to rapidly gain insight into the exposed attack surface of a system. The strong point of this method is the simplicity with which it is possible to gain a first indication of network security. Probably even more important, our method also permits a

comparison amongst the attack surface areas of different network zones.

However, some points require further investigation. Even if the proposed function $\mathcal{S}(.)$ gives a comparable and suitable estimation of the attack surface, the best function to use is an open problem, and we believe that mathematical analysis and statistics can greatly help in the choice of the optimal one.

The proposed approach gives a first rough (in)security measure, which increases with the number of available services. In this, we are encouraged by security tendencies in network management, which try to minimize the number of exposed ports, and, more generally, the number of exposed services. In spite of this, for a comprehensive evaluation schema, in addition to our assumptions several different adversary models must be considered, such as attackers acting directly on the host itself, firewalls affected by weaknesses (or simply misconfigured), and attacks to the information flow between hosts, that can expose information leakages or integrity violations.

In conclusion, although not a complete and perfect solution, this work is a step forward towards the definition of a much needed security metrics for networked ICT systems and can be used as a foundation for more complex and complete solutions.

ACKNOWLEDGEMENTS

This work is part of the POSITIF and DESEREC projects, funded by the EC under contracts IST-2002-002314 and IST-2004-026600

REFERENCES

Albrecht, A. J. and Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, 9(6):639–648.

Atzeni, A. and Lioy, A. (2005). Why to adopt a security metric? A brief survey. In *Proc. of QoP2005, First International Workshop on Quality of Protection, Milan (Italy)*, pages 1–12.

Budiarto, R., Sureswaran, R., Samsudin, A., and Noor, S. (2004). Development of penetration testing model for increasing network security. In *Proc. of Information and Communication Technologies: From Theory to Applications, Damascus (Syria)*, pages 563–564.

CC (2006). Common criteria for information technology security evaluation v3.1. [Online] http://www.commoncriteriaportal.org/public/consumer/index.php?menu=2.

Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–494.

Disney, A. and Johnson, P. M. (1998). Investigating data quality problems in the PSP. In *Proc. of the 6th ACM SIGSOFT international symposium on Foundations of software engineering, Lake Buena Vista (FL, USA)*, pages 143–152.

FIRST (2005). Common Vulnerability Scoring System (CVSS). [Online] http://www.first.org/cvss/cvss-guide.html.

fyodor@Insecure.org (1998). Nmap security scanner. [Online] http://www.insecure.org/nmap/.

Hauser, V. and Revmoon, D. J. (2006). The hacker's choice AMAP application mapper v5.2. [Online] http://thc.org/thc-amap/.

Howard, M., Pincus, J., and Wing, J. (2003). Measuring relative attack surfaces. In *Proc. of Workshop on Advanced Developments in Software and Systems Security, Taipei (Taiwan)*.

Howard, M., Pincus, J., and Wing, J. (2005). *Computer Security in the 21st Century*, chapter 8, pages 109–137. Springer.

Martin, B., Sullo, C., and Kouns, J. (2002). OSVDB: Open Source Vulnerability Database. [Online] http://www.osvdb.org/database-info.php.

McCabe, T. (1976). Complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320.

Microsoft (2003). Microsoft security alert severity matrix. [Online] http://www.microsoft.com/technet/security/alerts/matrix.mspx.

MITRE (2001). Common vulnerabilities and exposures web site. [Online] http://www.cve.mitre.org/.

Nicol, D., Sanders, W., and Trivedi, K. (2004). Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 1(1):48–65.

NIST (2005). National vulnerability database. [Online] http://nvd.nist.gov/.

SANS (2003). Critical vulnerability analysis. [Online] http://www.sans.org/newsletters/cva/.

Schudel, G. and Wood, B. (2000). Adversary work factor as a metric for information assurance. In *Proc. of New Security Paradigm Workshop, ACM/SIGSAC, Ballycotton (Ireland)*, pages 23–30.

SSECMM (2003). Systems security engineering capability maturity model v3. [Online] http://www.ssecmm.org/index.html.

US-CERT (2003). CERT vulnerability scoring system. [Online] http://www.kb.cert.org/vuls/html/fieldhelp#metric.