

# **A Single Metric for Evaluating Security Products**

*Dr. Igor Muttik  
McAfee Labs*

## **About author**

*Igor Muttik graduated from Moscow State University in 1985. His Ph.D. in 1989 was based on the research of semi- and super-conductors at low temperatures. He became interested in computer viruses in 1987 when PCs in the lab were infected with Cascade. In 1995 he joined Dr. Solomon's Software in the UK as a Virus Researcher. In 1999 he headed McAfee Avert in Europe. He speaks regularly at security conferences.*

*Dr. Igor Muttik currently holds a position of Senior Architect at McAfee Labs. He is also a member of the Board of Anti-Malware Testing Standards Organization (AMTSO). Igor lives in England with his wife Elena and 3 children.*

*Phone +44 1296 318700 Fax +44 1296 318722 Email: mig@mcafee.com*

## **Keywords**

*Security, protection, metric, malware, vulnerability, timing, attack history, integral, security reaction, cloud*

## Abstract

*We discuss the limitations of the traditional “binary” protection approach (“detected/missed” = “protected/unprotected” = “success/failure”). We show examples of how the growing frequency of attacks dictates a statistical approach to measuring the quality of security software. We analyze factors contributing to the probability of successful protection, present the mathematical approach to calculating this probability, and discuss how this can be implemented in practice.*

*For each attack, the “success of protection” is a function of time. For multiple attacks, we will have a set of such functions. We argue that a simple and meaningful numeric representation of this set of functions is a probability calculated and based on the integrals of these functions over time.*

*In this model, overall probability depends on the timeframes used to evaluate each attack. To be meaningful, the selection of these timeframes has to take into account users’ exposure to the threat. But full knowledge about the exposure is available only after the attack, so we have to deal with historical data.*

## Introduction

The purpose of security software is to minimize the cost of computer ownership. Ideally, security should protect a computer from the effects of any malicious attack while staying completely invisible. Historically, the assessment of successes and failures of security products was based on using simple binary logic—a computer was either "protected" or "unprotected" because a corresponding malware sample was either "detected" or "missed." The majority of tests that compare anti-virus (AV) software are based on simply counting the misses over a set of files.

There is a common understanding within the industry that the testing of security products should improve—this was reflected in the formation of the Anti-Malware Testing Standards Organization, or AMTSO (<http://www.amtso.org>). AMTSO developed a set of jointly agreed principles and guidelines that give advice for improving testing. They do not yet, however, describe a single metric for evaluating the quality of protection provided by security products. In this paper we present an approach to defining such a metric.

An important factor that forces us to give up the binary approach to measuring security software quality is the proliferation of “cloud-based” security solutions. By cloud-based we mean any computer protection technology that actively communicates with external servers (usually Internet based). Cloud-based security has an ability to deliver protection so quickly that the difference between reactive and proactive solutions almost disappears. The deployment of security updates is becoming almost instant on the global scale.

## Related research

Several papers described the mathematics related to deploying updates in relation to worm containment (Vojnovic & Ganesh, 2005; Xie, Song & Zhu, 2008). These models deal with the notions like “percentage of protected computers” and track this value over time. They are focused, however, on the mechanics and speed of deployment for a single software update, based on traditional software patching approach.

The concept of “probability of successful attack” and “probability of protection” was used by Edge (2007) to develop a metric based on attack and protection trees. This work, however, does not deal with the timing of protection. And the timing is a crucial element of the quality of protection.

There is a NIST publication (Mell, Bergeron & Henning, 2005) which defines three main categories of patch and vulnerability metrics: susceptibility to attack, mitigation response time, and cost. This paper (which is essentially a process guideline document), however, only enumerates all the contributing factors and do not provide any specific use case scenarios and does not give advice on how to compute meaningful aggregate metric scores.

Apart from these research efforts much has been done on a practical front, as part of anti-malware testing. Such tests are traditionally based on “success/failure” approach.

## Practical problems with the “success/failure” approach

When computer users see comparative tests specifying detection rates (usually presented as a percentage over a test set), they may unconsciously interpret these values as the probability of successful protection. For example, a 100 percent detection rate would be generally assumed to provide perfect protection, while a product with 90 percent detection rate would fail to protect the system 10 percent of the time. What users may fail to realize is that commonly available test

results are usually based on known malware samples and thus provide scores skewed towards “reactive” protection. In principle, there could be a product that fails miserably in the field but scores really well in a test—if the detections are added right before the test starts.

Apart from “reactive tests,” other tests try to isolate and measure the “proactive” capabilities of scanners. These show much lower detection scores—for example, check “Retrospective/Proactive” detection rates vs “On-Demand Comparative” (AV-Comparatives, 2004-2009). Some tests attempt to mix reactive and proactive results together (Hawes, 2009).

You might think that the real probability of successful protection lies somewhere between the numbers obtained in reactive and proactive tests. That’s logical, isn’t it? Unfortunately, the gap is really wide. How useful would be a statement that real protection probability is between, say, 30% and 99% boundaries? Additionally, even separating pure reactive and proactive scores is pretty much impossible so the boundaries are rather fuzzy. All products have both reactive and proactive capabilities; they employ many protection techniques - not just AV scanners. Moreover, even pure AV scanners normally have built-in, updatable heuristic and generic malware recognition. Thus they provide fairly agile proactive protection.

Traditional methods of measuring proactive protection use a “retrospective” approach - a frozen product (one that is not receiving updates) is tested against the malware that appeared after the freeze point. However, the duration of this freeze can have a dramatic effect on the results. It is easy to imagine a security product that very quickly reacts to threats in the field and updates its heuristics and generics. This is not a theoretical speculation—there are now security products that employ anti-spam rules, URL and IP blacklisting, etc. These rules are frequently updated and they block malware, too. A security product can proactively catch a newly spammed piece of malware just a few minutes after receiving a fresh anti-spam rule. And, of course, any retrospective test that froze the product before that rule was received would register the malware sample as not proactively detected. Yet in the real world it would have been.

An even more complicated situation occurs when the protection is not delivered incrementally to the client but is either constantly streamed or is cloud based. Such products just cannot be tested with the frozen definitions (cloud is external and not under tester’s control). Plus, products sometimes use combinations of these updating methods. As we saw in the anti-spam rule example, the timing of the protection delivery becomes very important for the test result to be correct. It almost seems that the tester must know how the product works to test it properly.

The AMTSO guidelines on cloud-based security products reflect some of the realities we’ve already described. This document recommends using a statistical approach when performing comparative tests of cloud-based security solutions. This method requires collecting field data over time and averaging the results.

Imagine a product that always misses the first attack but always detects the second (or subsequent) one. Such a product would score zero in proactive protection, but overall it would actually provide very good protection to most of the users. This is clearly a problem with retrospective testing of proactive protection and with the “binary” detection approach.

Yet another testing metric of the AV product’s quality is tracking the time between the first sighting of a threat and the moment when protection is made available (Marx 2004a, 2004b). This “reaction time” testing was born when there were global outbreaks in 1998-2003. Global outbreaks are now the thing of the past and so the popularity of this kind of testing dropped. The “reaction time” approach, however, demonstrates the high importance of the timing factor in evaluating security products.

## **Contemporary malware attacks**

Contemporary malware distribution occurs in waves. The bad guys are now largely driven by monetary incentives. Once their returns from a piece of malware start to diminish, they launch a new attack. In our opinion, a very important aspect of evaluating protection is switching from samples-based testing to attack-based testing. By an “attack” we mean the distribution of the same piece of malware over a period of time.

We can view malware infections as analogous with real life: imagine, for example, injecting a virus into a guinea pig and checking to see if the animal falls ill. In the short term, the virus is likely to replicate a few times, which causes the immune reaction and production of antibodies. They find and kill the viral copies so our guinea pig is again healthy. However, there is an immune system reaction to the virus—a short learning process that occurs before a reliable response is deployed.

Security products operate similarly: they produce a response to the new attack and deploy it. They can observe a piece of malware just once or twice and protect many millions of the users after that point. This response becomes especially important with cloud-based security, where global online threat intelligence delivers data about new attacks almost

instantly and the deployment of the response is also global and immediate. In this scenario, even a 100 percent reactive solution could be extremely effective in protecting users globally. After just a handful of reports about an attack, all other users are protected. For these protected users the “reactive” security product provided proactive protection! Thus proactive and reactive approaches are inseparable. Moreover, the reaction time plays a big part in converting one into another. Consequently, we wonder if there is any reason to ever test proactive and reactive properties separately. Essentially, we see that there is no way to say at any given moment whether a product’s protection is reactive or proactive. But what we **can** say is whether the protection is available or not.

As usually happens, understanding a problem in detail logically leads you to a solution. In this case we propose to perform continuous testing over time to accumulate security responses. Then we’ll use this recorded data to compute the probability of protection by a given product—regardless whether that protection is reactive or proactive.

### **Suggested metric: the probability of successful protection**

The timing of providing protection plays the most important role in the probability of successful protection. To track protection over time we need to monitor the security response and do it not just once (as it is done in current “reactive” and “proactive” testing models).

An additional benefit of continuous testing is that products may actually temporarily lose detections during attacks (and both reactive testing and retrospective testing are very likely to miss this fact altogether). In theory, a product may have unreliable detection in principle (for example, due to a memory footprint issue, uninitialized variables, or something similar), and the overall detection score observed in a test may vary considerably from the real one.

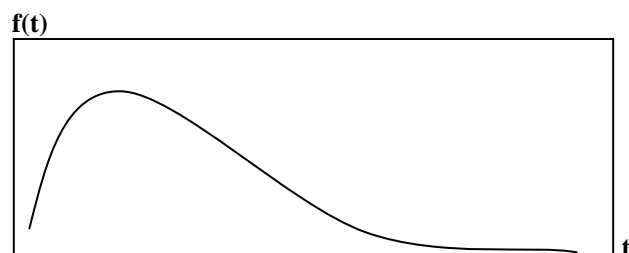
So we bring together the reactive, proactive, and response-time metrics and suggest tracking the security reaction of the product over the time of the attack. Later, we’ll discuss how to combine the results obtained for individual attacks into a sensible overall score. But let’s first look at tracking an individual attack.

All attacks have a starting point, which is when the first computer receives a piece of malware (regardless of how it arrives). This could also be a pointer to malware (such as a URL). To track this specific attack going forward it is important to capture enough data to classify subsequent attacks as “the same” or “different.” This is easy to do for static malware (you can simply compare samples or their strong cryptographic hashes) but can be hard in the case of polymorphic code (regardless of whether it is self-morphing or server-side polymorphic). It is important to realize that at this stage we cannot rely on security products to classify or group attacks because the protection may not be there. This is fine—we need only record the product’s reaction. Whether security reaction is correct can be evaluated when the whole attack history is available. Once we have full historical data we can extract the intrusions corresponding to the same attack thus splitting the data into individual attacks. This is not a trivial task and would require special tools. (One could rely on security products but this is, of course, not a good solution as the products under test should not be used to manipulate the test data. In the next section of our paper we shall discuss how to avoid this “attack-splitting” step.)

After the attack is finished we have a graph of its intensity as a function of time (assuming we have enough data points, of course—that depends on the number of honeypots/sensors/reports/etc. in the field). Depending on the type of the threat and scale of the attack (global or targeted, for example), the graph would look very different:

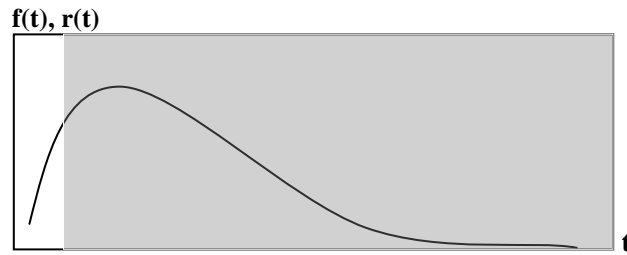
1. For self-replicating threats, the attacks may subside very slowly. This depends on changes in the level of malware reproduction in its ecosystem.
2. For mass-spammed malware, the initial uptake may be very rapid.
3. For non-replicating malware, attacks would likely be shorter and would normally be fairly quickly replaced by different attacks.

Here is an example of how an attack-intensity recording may look:



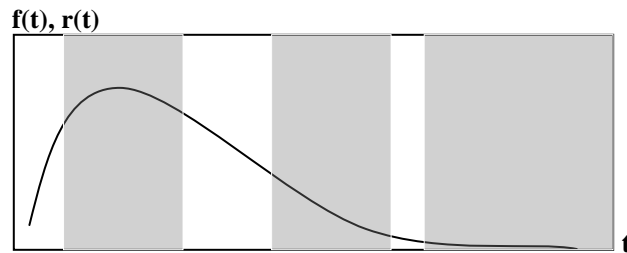
**Figure 1. A graph of field sightings vs time**

Now, let us superimpose the security reaction over that attack graph (gray area indicates that a product had protection, i.e.  $r(t)=1$ ):



**Figure 2. Field sightings  $f(t)$  and security reaction  $r(t)$  in gray**

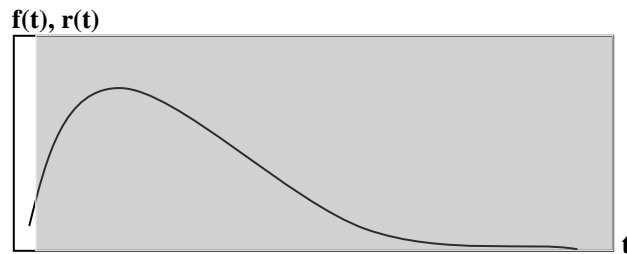
Or, if a product, for example, had unreliable security reaction then we may have a graph like this:



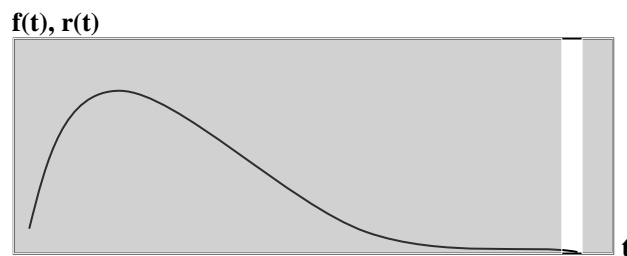
**Figure 3. Unreliable security reaction  $r(t)$**

Several simple metrics can describe the quality of the protection presented in these graphs: the delay in providing the first reaction (same as the reaction time by Marx, 2004), the reliability of protection (“unreliable” if it was ever lost after being first introduced), etc.

We argue that it is necessary to integrate the factor of the attack-intensity function and of protection to obtain sensible results. Have a look at the following two examples in Fig.4-5:



**Figure 4. Security reaction  $r(t)$  missing attacks at start**



**Figure 5. Security reaction  $r(t)$  missing attack at the end**

The first provides no proactive protection but successfully covers the great majority of users. The second product may be labeled unreliable but the temporary failure at the end of the malware attack would also affect only a small number of users. Neither is perfect. But a “protection gap” of the same duration during the attack peak would have been much worse! How can we make sure these factors are taken into account in our metric?

A formula to calculate the protection probability (quality of protection) for an attack would look like this:

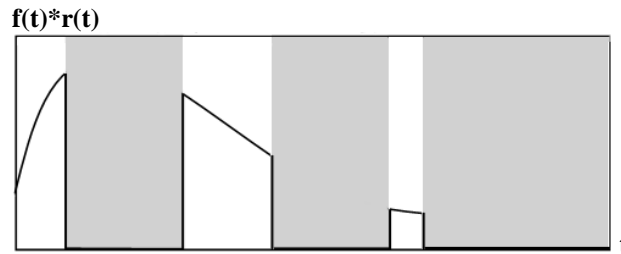
$$p = \int f(t) * r(t) dt / \int f(t) dt$$

**Equation 1. Probability of protection for an attack**

In our formula  $f(t)$  is the attack intensity and  $r(t)$  is the security reaction (both are functions of time). If  $r(t)=1$  (protection always available), then we have a trivial case of dividing equal integrals and the resulting  $p=1$  (protection during the entire attack was always perfect, or 100 percent). You can see that our approach gives more weight to security failures when the attack is most intensive and affects most users—exactly as it should be.

It is a trivial fact that the probability of successful protection “ $p$ ” defines also the probability of successful attack which is simply  $(1-p)$ .

When we superimpose the security response and the attack frequency we get a function which represents the “population exposure”. To some extent this approach is a generalization of the term “zero day” – we essentially replace the binary decision (“zero-day” or “not zero-day”) with analyzing the exposure function for each attack.



**Figure 6. Exposure function –  $f(t)*r(t)$**

Our metric can also be applied to the calculation of the return-on-investment (ROI) values for both defenders and attackers (although the latter, of course, is an unfortunate side effect!). For example, if “ $p$ ” (the security reaction) is very low then the attackers have a high ROI. If a security vendor has growing “ $p$ ” then the R&D investments do actually result in increasing protection of their users.

## Multiple attacks

If we deal with many attacks, then we may wish to come up with a total score covering them all.

$$p = \sum_{i=1..N} ( \int f_i(t) * r_i(t) dt ) / \sum_{i=1..N} ( \int f_i(t) dt )$$

**Equation 2. Probability of protection for multiple attacks**

In this case  $N$  equals the number of attacks. It is logical to assign more importance to common field attacks and this, fortunately, happens automatically in our model because the integrals of “small” attacks would have a lesser impact on the divisor sum.

We can also add weighting into the summing above and give more weight to “more dangerous” attacks (for example, those stealing user data). Each weight may, of course, reflect the cost of an unmitigated attack. It has to be noted that we do not need to give any additional weight to different attacks due to their field prevalence because (assuming that the field data for all attacks is coming from the same network of sensors) relative scale of the attacks is automatically taken into account.

But can we simplify the calculations and remove the step of separating the attacks? Yes, but only if we treat all field sightings of malware as equal (that is, if we do not assign different weights/costs to different attacks).

If we do that then the computations in Equation 2 are reduced to the same formula as in Equation 1 which we used for a single attack. Essentially, all field sightings from multiple attacks are treated as **one** attack with no internal structure.

## Practical implementation

The theory above is nice, but we must eliminate several obstacles before this idea can be used in practice:

1. Not having enough field sensors/honeypots/traps/reports may not allow decent tracking of the attack-intensity function  $f(t)$ . Unfortunately, there is no substitute for real field data, so the only proper way to solve this problem is get more field feeds. There is an industry effort underway for sharing sample meta-data in IEEE XML format (IEEE, 2009). This effort could assist in boosting the volume of field data.
2. There is no clear definition of  $r(t)$ . Whether a security product is successful in providing protection may be debatable because some products (typically behavior-based products) may, for example, react after executing malware. Some malicious actions may have occurred at this point; thus evaluating whether blocking is successful (no stolen user information? no data exgress? no persistent changes to the system?) could be controversial. All in all, defining  $r(t)$  is not a trivial task and may justify writing a separate paper. (AMTSO is expected to publish guidelines about this soon, so watch for <http://www.amtso.org/documents.html>.)
3. We have to deal with discrete summing (instead of integrals) because the real  $f(t)$  and  $r(t)$  are not going to be mathematical functions but most likely timed records in a database. This is trivial to do:

$$p = \frac{\sum_{i=1..N} (f(t_i) * r(t_i))}{\sum_{i=1..N} f(t_i)}$$

**Equation 3. Practical calculation - discrete sums instead of integrals**

Here  $N$  is the total number of time points  $t_i$ .

There is an obvious optimization - we can exclude time points when there were no updates to security software (which means  $r(t)$  is the same as it was before). But this optimization would not work though with cloud-based security products.

Additional considerations:

1. If security software uses different update cycles for different components (for example, one for reactive protection and another for proactive), then the calculation of the final probability needs to cover periods longer than the update cycle to perform meaningful integration and averaging.
2. The generation of local knowledge (learning through artificial intelligence, which is capable of creating a local security response).
3. For cloud-based security there could easily be a dependency on the location of the client and the connectivity conditions. The frequency of updating is also unknown, so all field sightings would require a check.
4. If we treat all attacks as **one**, we can no longer define the “start” and “finish” times. The selection of these times may affect the results (especially if the selection falls at the “start” of an intense malware attack).
5. Only when there are field sightings of malware should we verify the protection function  $r(t)$ —because when the attack is in progress any temporary glitch in protection should affect the quality of protection. At the same time, it would be wrong to check protection when there are no field sightings because protection failures at these “quiet” times would have no effect on users in the field. (We assume, of course, that the traps/honeypots/reports provide adequate field visibility.)

## Conclusions

We demonstrated with examples that evaluating contemporary security products requires the tracking of attacks and protection over time.

Our suggested method of calculating the probability of successful protection against an individual attack provides a sensible coverage for these attack scenarios and types of protection (reactive, proactive, or unreliable). Therefore, this single metric can be used to evaluate and compare different kinds of security products, including even very hard-to-test, cloud-based security solutions.

The metric we described can be applied to evaluating the quality of patching as well as software updates in general.

## **Acknowledgements**

I want to thank my wife Elena and my children for putting up with my long hours at work for so many years. I also would like to thank God for supporting this research ☺

## References

- AV-Comparatives' "Retrospective/ProActive Tests" for 2004-2009  
<http://www.av-comparatives.org/comparativesreviews/main-tests> and  
[http://www.av-comparatives.org/index.php?option=com\\_content&view=article&id=127&Itemid=162](http://www.av-comparatives.org/index.php?option=com_content&view=article&id=127&Itemid=162)
- K.Edge "A Framework for Analyzing and Mitigating the Vulnerabilities of Complex Systems via Attack and Protection Trees"  
<http://handle.dtic.mil/100.2/ADA472310>
- J.Hawes "VB RAP Testing"  
<http://www.virusbtn.com/vb100/vb200902-RAP-tests> and  
[http://www.virusbtn.com/news/2009/10\\_09.xml?rss](http://www.virusbtn.com/news/2009/10_09.xml?rss)
- "IEEE Meta-Data Sharing XML Schema"  
<http://grouper.ieee.org/groups/malware/malwg/Schema1.1/>
- A.Marx "Outbreak Response Times: Putting AV to the Test"  
[http://www.av-test.org/down/papers/2004-02\\_vb\\_outbreak.pdf](http://www.av-test.org/down/papers/2004-02_vb_outbreak.pdf)
- A.Marx "Antivirus outbreak response testing and impact"  
<http://www.virusbtn.com/conference/vb2004/abstracts/amarx.xml>
- P.Mell, T.Bergeron, D.Henning "Creating a Patch and Vulnerability Management Program"  
<http://csrc.nist.gov/publications/nistpubs/800-40-Ver2/SP800-40v2.pdf> p.1-2
- M.Vojnovic, A.Ganesh "On the Effectiveness of Automatic Patching"  
<http://www1.cs.columbia.edu/~angelos/worm05/patch.pdf>
- L.Xie, H.Song, S.Zhu "On the Effectiveness of Internal Patching against File-sharing Worms"  
<http://faculty.frostburg.edu/cosc/hsong/papers/acns08-worm.pdf>