

# Malicious Test Day

Affectionately  
Wreaking Havoc

Ted F. Rivera  
Adam Tate  
Scott A. Will

March 1, 2003

# Table of Contents

<a href="#">Table of Contents</a> .....	ii
<a href="#">Introduction</a> .....	1
<a href="#">Malice – in Measured Doses</a> .....	2
<a href="#">Administering a Malicious Test Day</a> .....	3
<a href="#">The Unexpected Challenge of Scheduling</a> .....	4
<a href="#">Involving Others and Some of the Benefits It Affords</a> .....	4
<a href="#">Ownership</a> .....	5
<a href="#">Learning from Others</a> .....	6
<a href="#">Is Once Enough? How Many Are Too Many?</a> .....	8
<a href="#">Change is Good: Fostering Creativity</a> .....	8
<a href="#">Evil Intent with a Loving Purpose</a> .....	9
<a href="#">Author Biographies</a> .....	10
<a href="#">Ted Rivera</a> .....	10
<a href="#">Adam Tate</a> .....	10
<a href="#">Scott Will</a> .....	10



## Introduction

Being a tester is a challenging occupation, not only because of the nature of the work itself, but also because of the attitude and perspective it often demands. Ron Patton, currently a testing consultant and formerly a test manager overseeing some very well-known products, writes:

Your job is to inspect and critique your peer's work, find problems with it, and publicize what you've found. Ouch! You won't win a popularity contest doing this job.<sup>1</sup>

It is often the case in the books and articles written about software testing that subjects like *politics* are prominently featured, and the guidance offered to the tester is often presented as if he must somehow become a skilled negotiator among the larger organization in order to find enduring success.



At times, this emphasis can create an atmosphere where the test team views itself as if walking on the eggshells of the development landscape – being careful to watch every step. It is interesting to note that the chapter in Rex Black's book, *Managing the Testing Process*, entitled “The Triumph of Politics: Organizational Challenges for Test Managers” is longer than the chapter “Plotting and Presenting Your Course: The Test Plan.”<sup>2</sup> He writes, “[A]s a manager, you must be sensitive to political realities, some of which are unique to test management.”<sup>3</sup> This emphasis, while often muted and subtle, is real.

In the introduction to the chapter entitled “Thinking Like a Tester,” Kaner, Bach, and Pettichord offer the following perspective:


Testers come from many backgrounds. They are a diverse bunch, but most people agree: Testers think differently. How do they think differently? Some say testers are “negative”

---

<sup>1</sup> Ron Patton, *Software Testing*, SAMS, Indianapolis, Indiana, 2001, p. 45.

<sup>2</sup> Rex Black, *Managing the Testing Process*, Wiley, New York, 2002.

<sup>3</sup> *Ibid.*, p. 333.



thinkers. Testers, they say, like to complain and break things, and take a special thrill in delivering bad news. This is a common view. We propose an alternative view. Testers don't complain; they offer evidence. Testers don't like to break things; they like to dispel the illusion that things work. Testers don't enjoy giving bad news; they enjoy freeing their clients from the thrall of false belief. Our view is that thinking like a tester means practicing epistemology. Testing is applied epistemology, not grumpistics or whinography.<sup>4</sup> [Underscore added]

This is a very positive and productive view of the profession of testing – one we agree with and support wholeheartedly. At the same time, take a look at what we've underlined above.

Testers are people like everyone else. We succumb to daily ruts, we develop comfortable, predictable habits and patterns of work and practice. To overcome these tendencies, and in a spirit of love and cooperation, with an aim of mutual benefit, we propose the following: every now and again, we should allow a wholly different attitude to prevail. "The Grinch got an idea, an awful idea. The Grinch got *a wonderful awful idea*."<sup>5</sup>

While it may be wise and prudent as our normal course of action to humbly navigate the landscape of the software engineering community, Malicious Test Day affords the rapturous possibility that there should come a time when we unshackle our minds from its normal constraints, a day on which we give full vent to the notion that it's absolutely liberating on the most infrequent occasions to *relish* breaking things and to *thrill* in delivering bad news. And thus, *Malicious Test Day* was born.<sup>6</sup> And so, with a tip of the hat to Admiral David Glasgow Farragut, "Damn the politics – full speed ahead!"

## Malice – in Measured Doses



So, what exactly *is* Malicious Test Day? Malicious Test Day is a day that we set aside, generally once a quarter, where the entire System Test and Quality Assurance organization devotes an entire day to testing an integrated product

---

<sup>4</sup> Cem Kaner, James Bach and Bret Pettichord, *Lessons Learned in Software Testing*, Wiley, New York, 2002, p. 11.

<sup>5</sup> Dr. Seuss, *How the Grinch Stole Christmas*, Random House, New York, 1957.

<sup>6</sup> It is at this point you should hear an evil-sounding cackle reverberating in the inner recesses of your mind...



suite. Most of our products are designed to work together, so we simply install our products across multiple environments, with different operating systems and databases, create special user IDs for each member of the organization, and tell them to “Have at it!” There are two parallel emphases during a Malicious Test Day. The first concentration is for those who wish to have no step-by-step instructions given, no mandated tests to perform. The only guidance we give to these participants is:

Play with the products – try to break them if you can. If you come across something that even remotely looks like a defect, or if you see a performance problem, or any other anomaly, write up a defect report. If you see something that can be improved, write up an enhancement request.


Needless to say, this aspect of Malicious Test Day is rather “free-form” in its approach, but this type of testing can sometimes mimic customers’ usage of the products in a way that highly-structured tests sometimes cannot. Imagine 50 or 60 people, all using various aspects of your product suite at the same time, and all thinking of devilishly clever ways to break something. We’ve found it to be quite fun, and quite a learning experience as well! That being said, however, we generally also have some comprehensive and complex scenarios in mind that can only be executed with significant coordination beforehand.

## Administering a Malicious Test Day

The principal objective of this event, of course, is to shake ourselves out of the common routines of our testing and look with new eyes at the software under development. It is an opportunity to answer questions such as these, as well as many others:

- How does a *new user* look at this product?
- What might *frustrate* experienced users?
- What might a *hacker* attempt?
- What kinds of *security exposures* might exist?
- What areas of the product am I *unfamiliar* with?
- What can I do to *test the limits* of this product?
- What *environmental oddities* can we create?
- What have we *never tried before*?

*What questions should you be asking?*



To be honest, we thought running a malicious test day would be a breeze – just pick a day and let glorious mayhem flow. To our surprise, though, we found that planning fruitful malice is hard work.

What follows are several areas that we have found that we need to pay attention to prior to executing a successful Malicious Test Day. Review the following considerations well in advance of implementing your Malicious Test Day in order to orchestrate a truly productive event.

## The Unexpected Challenge of Scheduling



A Malicious Test Day can be as simple or as complex as you want it to be. Our approach was somewhat involved, as we were looking for benefits beyond the obvious. As a result, timing the event became an important consideration. Depending on the size of your company, this may present no small challenge.

First, we wanted a timeframe that was just after major releases of one kind or another for the majority of our test teams – the worst mistake you can make is to schedule an event such as this when the team is already pulling overtime to close out a release. In addition, we tried to avoid low attendance periods (such as peak summer months or late-year holidays). Further, we took into account activities in other organizations that might conflict, in order to maximize participation from groups outside of the test teams. We then publicized our event as early as possible and obtained the commitment of key leaders.

## Involving Others and Some of the Benefits It Affords

Probably the principal advantage to such advance planning is that we were able to get the commitment of other groups to participate fully. Consider the possibilities. What if you were to invite...

- *The development team.* Do you think they might enjoy testing another group's product? Writing code to overwhelm it and expose weaknesses? Might they want to test the edges of their own product? To prove the superiority of the code they wrote? As an added bonus, participation by the development team in our Malicious Test Day activities provides you with an opportunity to showcase ways in which heightened ongoing involvement and





interaction with your test team might be both possible and mutually beneficial.

- *Sales, information development, marketing, and services.* These groups often are able to bring in a client perspective that more inwardly-oriented groups might miss. In addition, in many cases, this gives these groups an opportunity for hands-on experience and interaction in a safe atmosphere – all questions are welcome. We have found Malicious Test Days to be profitable means of education for many groups.
- *Other test teams.* By engaging other test teams, you gain an additional professional perspective, but also, they gain the opportunity to better understand how your product set works and, potentially, how it interacts with the products they test.

## Ownership



There are some people who treat rental cars as gingerly as they would their own. They check the oil, keep the interior in pristine condition, run it through the car wash, and use the accelerator with only measured enthusiasm. In general though, car *owners* are far more accountable for the maintenance and care of their vehicles than car *renters*. Malicious Test Days function best when someone knows this is *their event* to own and run. They must have the latitude to customize the event, to inject fun, and to try new things.

There are some constants in the equation which the owners of your event will need to keep in mind:

- *Event Setup*

We have found that by far the most time-consuming aspect of our Malicious Test Days has taken place behind the scenes, setting up hardware, software, userids, permissions, and physical locations that allow for appropriate levels of interaction and collaboration. If it is your aim to involve people from a wide range of other organizations then there is no winging this – this takes a lot of hard work. It also takes planning to arrange the appropriate trinkets and trophies! More on this aspect below...



- *Coordinating the Day's Schedule*

As mentioned earlier, we have found it helpful to focus on products and combinations of products in an orchestrated manner, moving essentially from one major task to the next. In parallel, we also encourage those who have an interest in acting maliciously on their own, but perhaps the best results have been obtained from the “coordinated mayhem.” We pull together sessions we call “Über Tests,” where we have a large number of people hammer a particular product or scenario mercilessly. It is challenging to allow for both *planned* and *unscripted* activities, but it can be done!

And of course, it is essential to know when and where the pizza and goodies will be available!

- *Communication*

Prior to the event, a good deal of communication needs to take place in order to ensure that adequate representation is obtained from within and without the test community. On event day, people need to feel connected and must be able to interact – they need to know what is coming next and what defects are being discovered. Tote boards, bells, instant messaging, websites, and all kinds of other means of communication are useful ways of keeping those destructive juices flowing.

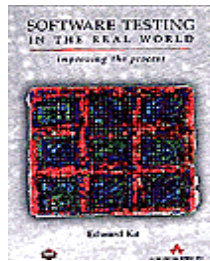
- *Representing client environments*

In addition to the individual environments we provide for the event, we also customize a number of scenarios that involve using real customer data (and if possible, real customers!). Recreating high numbers of users using real customer data produces real world high stress conditions (if planned and orchestrated appropriately, that is). In short, the owners of our event work with the test team to devise complex situations that they believe will push the products in ways they've never been pushed before.

## **Learning from Others**

We have found two books especially helpful for the perspective and insight they bring to bear on the mindset that a Malicious Test Day aims at. Edward

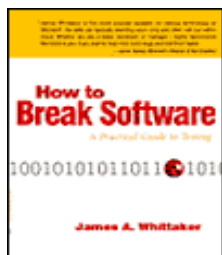




Kit's work on "Creative Destruction" is often cited by those who are familiar with his book, *Software Testing in the Real World*. He writes,

Testing is a positive and creative effort of destruction. It takes imagination, persistence and a strong sense of mission to systematically locate the weaknesses in a complex structure and to demonstrate its failures....What we need is someone else who can attack it with the attitude of: "I'm here to destroy this thing. I'm going to find the defects that I know are there; that's my job and that's what I'm paid to do."<sup>7</sup>

The chapter from which this quote is taken, "Good Testers Have a Testing Attitude," is solid background reading in preparation for the event, and will be beneficial for participants who have a background in testing or otherwise.



The second book which we have found especially useful for our Malicious Test Days is a veritable handbook for the event. It is James Whittaker's *How to Break Software*. Whittaker offers dozens of tips and techniques to drive out defects, and it is notable that he speaks specifically about events similar to Malicious Test Days which he refers to as "Bug Hunts," which are limited to two-hour events. He writes,

We hold bug hunts immediately after a major new build occurs or a new feature is added. We generally limit the hunt to a specific area of the software. So we look for clues from developers about what features have had the most code modifications or new code added, and we target those.

The purpose of a hunt is not only to shake some good bugs out of a new build but also to foster teamwork and friendly, healthy competition among your testers. We always hand out cash prizes and try our best to bow to the victors when we pass them in the hallways. Winners usually get the "we're not worthy" bow for a couple of days after a hunt.<sup>8</sup>

And so while the intent of Whittaker's bug hunt is a bit different than one of our Malicious Test Days, there are many similarities, and there are a number of excellent insights that can be gleaned from Whittaker's book as a whole for planning your own event. A good example is the manipulation of environmental characteristics (such as memory, disk space, etc.) that can cause an application no end of angina.

---

<sup>7</sup> Edward Kit, *Software Testing in the Real World*, Addison-Wesley, 1995.

<sup>8</sup> James Whittaker, *How to Break Software*, Addison Wesley, 2003, p. 124.

## Is Once Enough? How Many Are Too Many?

Given the approach we take for our Malicious Test Days, we decided to ask our test teams how frequently they should be held. Given the work involved and the setup required, we expected them to suggest annual or semi-annual events. To our surprise, they strongly suggested that they be held on a quarterly basis – the educational benefits, the actual defects discovered, and the opportunity to try new things warrants the investment of time. And did we mention these can be fun?

## Change is Good: Fostering Creativity

In order to ensure your Malicious Test Day leaves a mark well beyond the event itself, bear in mind three essentials to the festivities:



1. *Buy trophies*

We usually buy several trophies of varying sizes, with the largest reserved for the worst defect discovered. Accept no substitutes: find a trophy store that will stick a bug on the top of the biggest trophy they offer!

2. *Give rewards*

Many people work hard to make a Malicious Test Day successful. Recognize your owners, those who have helped behind the scenes, and those who have done the most damage, er, found the most bugs!

3. *Foster creativity*

Use Malicious Test Days as one of many organizational tools that can help to instill in your teams the notion that creativity in their work does bear real fruit. By thinking about testing in new ways, we can help ensure better-tested and, therefore, higher-quality products, and help also to make our clients happier.



## Evil Intent with a Loving Purpose

On Malicious Test Days, we find dozens of defects. We gain new perspectives. We learn many new things. We have fun and build our teams. That should be benefit enough, right? Done right, a Malicious Test Day can have even more far-reaching consequences.

Not long ago, a major power outage affected the Northeastern United States.<sup>9</sup> The kinds of difficulties encountered in a power outage seem like they are largely incidental to software applications – if the computers aren't working, neither is the software! In point of fact, though, such outages can cause extreme conditions for many of the applications developed, conditions not unlike those that might be reproduced in a Malicious Test Day. Consider all of the perfect storms that your applications might encounter, and design scenarios with the worst case scenarios of the real world in mind.

While it is true that testers need to be team players, fundamentally, there is an important dimension of our work that can be too often muted when “getting along” becomes the chief objective – it is our job to drive defects out of our software – mercilessly. ☺

\* \* \* \* \*

We would appreciate any comments, suggestions, or feedback based on your implementation that you would be willing to offer. You can reach us via the following email addresses:

Ted Rivera:	<a href="mailto:trivera@us.ibm.com">trivera@us.ibm.com</a>
Adam Tate:	<a href="mailto:atate@us.ibm.com">atate@us.ibm.com</a>
Scott Will:	<a href="mailto:sawill@us.ibm.com">sawill@us.ibm.com</a>

Thank you – and we look forward to hearing from you!

---

<sup>9</sup> In August, 2003, to be exact. See the following news account: [http://news.com.com/2100-1011\\_3-5063997.html](http://news.com.com/2100-1011_3-5063997.html).



## **Author Biographies**

### **Ted Rivera**

Ted is a Product Development manager with IBM Corporation's Tivoli Systems Software division. Ted manages several departments that provide system verification, performance and scalability testing, integration testing, documentation, globalization, and a range of other services in support of products in Tivoli's portfolio. He has worked at IBM for over 20 years in a variety of programming, customer support, and management assignments.

### **Adam Tate**

Adam Tate is a Quality manager with IBM Corporation's Tivoli Systems Software division. His experiences as an engineer and as a manager give him a well-rounded view of organizational issues as they relate to Quality. His experiences in both start-up and enterprise companies provide him with a unique perspective of today's technology business environment. His commitment to customers has won him awards for his successful involvement in hundreds of customer situations.

### **Scott Will**

Scott is a Quality Assurance manager with IBM Corporation's Tivoli Systems Software division. In his 13 years with IBM, he has held management roles in software testing and quality assurance, team-lead roles in customer support, and has been chief programmer for several projects, programming in C, C++, and Java.