E-MAILS FROM AN OLD BUG TO A YOUNG ONE

MADE PUBLIC BY

BRUNO MAŃCZAK

E-mails from an old bug to a young one

Bruno Mańczak

This book is for sale at http://leanpub.com/bug_emails

This version was published on 2022-07-13

Contents

Introduction . . . . . **i**

Condolences **1**
E-mail 1 1
E-mail 2 2
E-mail 3 3

Value **6**
E-mail 4 6
E-mail 5 6
E-mail 6 6
E-mail 7 7

Tester **8**
E-mail 8 8
E-mail 9 9
E-mail 10 10
E-mail 11 11
E-mail 12 12

Automation **16**
E-mail 13 16
E-mail 14 16
E-mail 15 16
E-mail 16 16

Software Paper Cuts **17**
E-mail 17 17
E-mail 18 17

DevOps **18**
E-mail 19 18
E-mail 20 18
E-mail 21 18

Afterword **19**

Introduction

👋 Hey fellow humans!

The contents of the e-mails that I share with you come from a data breach on BUGF.org, the site of ***Bugs United Globally Foundation***, where the defects hosted their e-mail server.

I believe the emails are worth sharing with the wider software development and testing community, as they give valuable insights into the way defects think about us and how they hide away from us. So far, I managed to decode the first few emails. The process of doing so is time-consuming, as bugs used strong and creative encryption methods. Nothing too complex for an experienced tester interested in security, though.

This book will be updated on a regular basis, while I proceed with decrypting the rest of the leaked data. So far I decrypted around **15%** of data. The latest version of the book will always be available on Leanpub. I will also post updates on my blog https://blog.bugf.org/ and through e-mail newsletter You can join the newsletter here: https://bit.ly/bugf-newsletter.

I would like to learn what your opinion is about the book. Is it an interesting, hopefully at moments even fun read? Feel free to mail me at bruno@bugf.org or if you don't like the idea of exchanging messages with another human beeing you can share your feedback using the https://tally.so/r/nrRkvn form. Thanks in advance!

Have a ~~good~~ great read!

Bruno

Condolences

E-mail 1

From: **HR System** <autoreply-HR-system@bugf.org>

Date: Mon, 7 Apr 2014 at 8:23

Subject: ✅ Absence request has been approved.

To: Glitch McDamage <glitch.mcdamage@bugf.org>

Hello Glitch!

Your absence request has been approved.

Request

Absence type: Bereavement Leave

Dates: 2014-04-07 to 2014-04-07

Details

check here

Best Regards,

HR Team

E-mail 2

From: **Michael Pest** <michael.pest@bugf.org>

Date: Mon, 7 Apr 2014 at 8:31

Subject: Bereavement Leave

To: Glitch McDamage <glitch.mcdamage@bugf.org>

Dear Glitch,

I am truly sorry to hear of the loss of your father. Please accept my condolences.

Following the company policy, I accepted your bereavement leave of one day. Would you require additional days off, you can submit a request for unpaid leave.

–

Best regards,

Michael Pest

Manager, CRM Software, BUGF

E-mail 3

From: **Uncle Blackout** <blackout.mcdamage@bugf.org>

Date: Tue, 8 Apr 2014 at 13:27

Subject: My condolences

To: Glitch McDamage <glitch.mcdamage@bugf.org>

My dear Glitch,

I am deeply saddened to learn about the fix of your father, Reaper. He was one of the cruelest and most mischievous defects that I knew. His devotion to breaking software was admirable. Whenever my project had a touchpoint with his doing, I knew I could count on him. Together we caused a lot of devastation, and we drove multiple enemies crazy.

Please know that I'm here for you. With your father gone, I feel the responsibility to take over your mentoring. We need to train you so that you're better prepared for the ongoing war with our enemy. For obvious reasons, we can't meet IRL, but feel free to DM me, or we can schedule a 121 video call. I would like to learn more about your project and how you're exploiting it.

It's a good moment to reflect on your father's life and take a deeper dive into his unfortunate end. If we put emotions aside, we can notice that it gives us a few valuable insights. Reaper's viciousness was one of his best qualities. However, I believe this trait was also the reason that drove him to the fatal fate. The patch with the fix was applied by enemies in their systems all over the globe in just a few days. Let this be a lesson to both of us, that we need to find that perfect balance between our severity and detectability. The perfect slot to be in is one where we're able to cause maximum damage and be undetected for as long as possible. However, multiple defects forget that we do not want to cause too much damage so that even once we're detected the enemy thinks it's harmless to keep us alive. This way, a fix could spend years on the list of issues to

fix, because there will always be some greater threat to fix. And while the enemies knowingly delay implementing the fix, we can grow with every hour and feed on their hesitation, undisturbed. In your father's case, the enemy immediately realized the gravity of the threat that he posed, and he was eliminated in no time.

On one hand, it was marvelous that your father was able to implement himself in just 2 files. Such damage rarely can be introduced with so little code. But on the other hand, it means that when he was fixed with `96db902`, it required just 40 small changes in 2 files. A more experienced defect might have considered spreading among multiple files, even more modules or systems to prevent efficient debugging and quick fix. Remember that the harder it is to fix us, the less the enemy is willing to do it. After all, it's just a job for him. Let your fix be the dullest, longest, most boring task they can think of. Such tasks require planning, and our enemy so often appears to be terrible at it! A properly settled defect can live for years, even once the enemy detects us. Many defects consider this the best place to be in. If settled correctly, we can leverage that time to connect and make alliances with other bugs, to jointly cause even more havoc. But your father went full rogue, he didn't want to settle for little damage. No, Reaper wanted to bring the whole new world of possible exploits. This comes with a price, as the enemy sensed the urgency of the required fix, once he met your father. All defects must choose their strategy, each choice coming with a risk.

How the enemy have met your father? A dull audit! What a trivial mistake it was, to make the enemy think that it's worth conducting an audit! Too many defects wanted to help your father. Too many defects jumped on that project and eventually made the enemy realize that it has hidden defects. What a mistake! What a waste! They should have left your father alone. Once the enemy understands that something is off, he wants to fix it, especially in critical software! This was a dream project that your father worked on: open-source, critical, and underfunded. He could have stayed undetected for much longer! Let us remember that the enemy must

think that HE is in control. You must sacrifice your fellow defects in as many numbers as possible and as often as possible. Don't let the enemy suspect anything! Let them believe they fixed you and your fellows!

There is one final lesson we can take from your father: it comes from the story of his conception. He came to life at a perfect moment. The critical Pull Request was merged shortly before noon on New Year's Eve. Excellent timing, we must seek such opportunities as frequently as possible. You might not know this, but our enemies can get tired and distracted. The bigger the distraction and tiredness, the easier it is to conceive and hide one of us. You must learn all about the holidays and the schedule of your enemy. Our usual strategy would involve taking a benefit from Fridays, Weekends, Public Holidays, and so on. But you can go much deeper than that if you truly understand your enemy. Usually, it's easier to hide if the enemy works under pressure and with a tight schedule. This means you would want also to explore moments close to deadlines, key milestones, and all other points of time to which your enemy assigns some importance. Those are the moments when he gets distracted the most.

Be inspired by your father! I believe that with the right circumstances, you are also capable of awful things. But you need to plan a proper wrongdoings strategy and execute it without any mistakes.

Your affectionate uncle

Blackout

Value

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 4

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 5

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

Share a document?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 6

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 7

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

Tester

E-mail 8

From: **FixProtec Alerts** <fixprotection-alerts-noreply@bugf.org>

Date: Mon, 19 May 2014 at 09:01

Subject: FixProtection Alert!

To: Glitch McDamage <glitch.mcdamage@bugf.org>

Dear Glitch,

our FixProtection monitoring has detected a critical alert for the project that you're working on.

New job opening: **Software tester**

Take action now

This email was sent from a notification-only address that cannot accept incoming email. Please **do not reply** to this message.

E-mail 9

From: **Michael Pest** <michael.pest@bugf.org>

Date: Tue, 20 May 2014 at 10:22

Subject: Report request

To: Glitch McDamage <glitch.mcdamage@bugf.org>

I just noticed the FixProtection Alert on your project. We need to understand why the enemy has decided to hire a tester for this project.

We managed to keep the project without any formal testing for sooo long… What happened? Why did the enemy decide to change the strategy?

I doubt they will fill the position soon, but we need to act quickly anyway. I expect a formal report with a root cause analysis until Friday.

Please cc Regina Leafcutter on the report.

–

Best regards,

Michael Pest

Manager, CRM Software, BUGF

E-mail 10

From: **Grandpa Chrysaor** <chrysaor.earwig@bugf.org>

Date: Mon, 19 May 2014 at 10:51

Subject: Data

To: Glitch McDamage <glitch.mcdamage@bugf.org>

Dear Glitch,

I hope you are well. G'ma can't wait to hear how you're bringing destruction to our enemy. Please call her more often.

I uploaded the data you requested to BUGF drive: drive.bugf.org/d/c2VjcmV0LWRhdGE= There's a lot of interesting information there. We hope you'll bring it to bad use.

Love,

Grandpa

E-mail 11

From: **Auris Worm** <auris.worm@bugf.org>

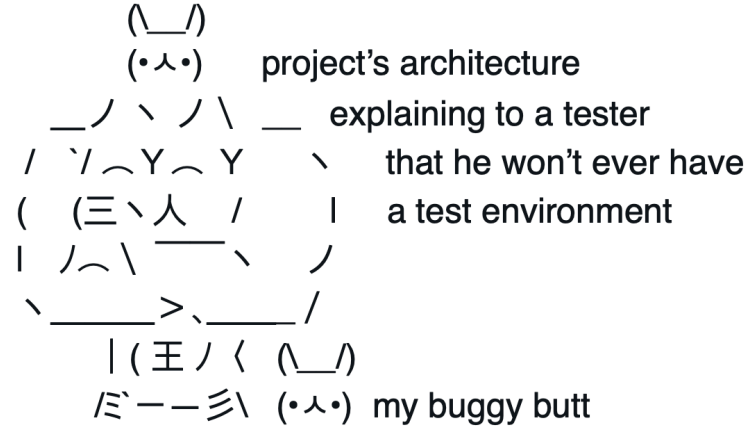
Date: Tue, 20 May 2014 at 9:23

Subject: Funny meme

To: Glitch McDamage <glitch.mcdamage@bugf.org>

Hey dude!

I've heard about the tester. Good luck with outsmarting that bastard!
Here's how we do it in my project:



E-mail 12

From: **Uncle Blackout** <blackout.mcdamage@bugf.org>

Date: Thu, 22 May 2014 at 19:23

Subject: Tester???!!!

To: Glitch McDamage <glitch.mcdamage@bugf.org>

My dear Glitch,

I note with great displeasure that your enemy decided to hire a tester. This should have not happened. You were too ostentatious with the induction of trivial defects, instead of focusing on slow but fruitful induction of something bigger. You made the enemy realize there is something wrong with how they develop the project. This forced them to decide to change the status quo and to try something different.

You're lucky that the enemy has not decided to take even more substantial steps to improve the project. From my observation, the simple hiring of a tester does not necessarily change anything in how the enemy works, given we guide the enemy properly. If the adversary already found many of us, without making any significant testing efforts, you must assure him that the current testing strategy is something that the new hire on the team should continue. Once the new antagonist arrives, make sure they find multiple meaningless defects. Now is the time to make sacrifices. Remember the alliances you made in the past: betray your allies and delete any links between you. Don't leave any trace of your involvement. Betray the visual bugs, let them be discovered by the enemy. Betray the typos, let them be fixed. Betray some happy path defects, to make the appearance of improving software quality. This way, in reality, you and your true (i.e. most devastating) allies will stay unfixed in the hidden parts of code.

Let the tester do the simplest of tests. If you guide them well, the enemy will be buried under the backlog of meaningless bugs to fix.

This way, some bugs betrayed by you will stay alive, so beware they don't grow with too much power, so they won't betray you and undermine your existence in the future.

If you're lucky, the tester might even bring more harm than good to the project. How? A badly involved tester could be seen as a person that offloads the developers by completing tasks done previously by them. This way, developers would reduce or even stop any significant testing on their side with an assumption that a tester would do it. Furthermore, a tester might assume that the developer already finished some more "technical" or "low level" tests. Consequently, new cracks might appear in the system: fewer unit tests, fewer white-box tests, more confusion, and more pushing tasks on others among your enemies. As a result, more room for us, defects!

You must prepare properly for the hide-and-seek game with your tester. He will become your primary attacker, he will be the face of your adversary that you see most often. However, remember that he is rarely the one that destroys you. Fix comes mostly from a developer, and it is the developer you need to fear the most. A tester is generally a mere whistleblower. His task is just to signal problems. He provides information about us to other stakeholders among the ranks of our enemy. The fuzzier this information is, the harder it is for the programmer to come up with his ultimate weapon against you. There are a few tactics you could apply to lead the tester around by the nose.

The easiest tactic is to bind your harmfulness to specific environmental conditions. Ideally, with conditions that do not appear on the test environment that your tester is using most of the time. Trivial and obvious approach, but works over and over again.

You could also update your evil works to not appear all the time. If the enemy repeats the same steps, but he's able to see you for example once in five attempts, then it's easy to overlook you or to assume that your appearance was random. Nobody knows how

to fix random. Random errors can be blamed even on cosmic rays breaking the electronics, leaving you intact.

Finally, add a few simple steps that would be required to induce you. The more work is needed to reproduce what the tester has said, the less likely it is that someone will try to repeat it. At the same time, it will be easier for enemies to make mistakes during the reproduction of those steps, as they frequently misunderstand each other. Our intelligence consistently reports that the enemy is terrible at communication. They are frequently unable to communicate effectively even with their loved ones, let alone their coworkers. Misunderstandings are in their nature.

I don't want my mail to become a book about best-performing tactics against our enemy. For that, go to our cyber library and read through one of our textbooks. You could also study our enemy's textbooks on how to develop good software. The irony is that they do know how to fight efficiently against us, yet they end up repeating the same mistakes over and over again. Not learning from the past, not learning from others, re-inventing the wheel instead.

Of course, a tester and more tests are a threat for us. The adversary knows that too. However, our enemies differ greatly in how they leverage this knowledge. Some decide to hire more people with the sole task of manual testing. Some decide to prolong the time needed to develop the project to account for tests. Others decide to start implementing test automation. The most dangerous enemy is the one who intentionally applies a mix of those tactics and ends up embedding the testing culture in every step of the development within the project. This is why simply hiring a tester does not pose much threat yet. Be aware of what kind of person is hired and what is expected to do. Let us hope this won't be the start of redesigning the whole development process.

In this mail, I just wanted to nudge you to update your destruction strategy slightly. You must choose for yourself what kind of approach you'll take. Watch closely the tester. Keep the greater bad

in your mind.

Your affectionate uncle

Blackout

Automation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 13

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 14

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 15

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 16

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

Software Paper Cuts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 17

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 18

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

DevOps

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 19

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 20

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

E-mail 21

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bug_emails.

Afterword

Thank you for reading!

New chapters will be added to this book at least monthly.

I would appreciate your feedback, preferably using https://tally.so/r/nrRkvn You can also mail me bruno@bugf.org

Stay healthy and take care!

Bruno