SDN MAGAZINE

Nummer
147
jun 2023

SDN Magazine is
een uitgave van:
SDN

Van en voor Microsoft & .NET professionals



> Will ChatGPT Replace Developers

> 5 manieren om bij te blijven als software developer

> Hacking video games with C#

SDN

Software Development Network

Word nu lid voor
maar €69,95 per jaar
en ontvang de volgende voordelen!

> **Gratis toegang tot Future Tech**
> **Gratis toegang tot SDN University sessions**
> **4 keer per jaar een gedrukt exemplaar van het SDN Magazine**

Voor elke serieuze.NET developer en/of bedrijven die Microsoft technologieën gebruiken is het een must om onderdeel te worden van de SDN. Als je up-to-date wilt blijven, is dit de manier om dat te doen.

Ga naar sdn.nl/lidworden/ voor meer informatie

Colofon

Uitgave
Software Development Network
26ste jaargang
Nr. 147 • juni 2023

Redactiecommissie:
Annejan Barelds, Nadine Wolff, Jan de Vries, John Bruin, Menno Jongerius, Marcel Meijer, Roy Janssen, Roelant Dieben, Thijs den Hollander.

Auteurs:
Richard van Zandwijk, Meara Leentvaar, Dennis Fatehmahomed, Frank van der Stam, Twan Koot, Carlo Kuip, Jasper Siegmund.

Listings:
Zie de website www.sdn.nl voor eventuele source files uit deze uitgave

Contact:
Software Development Network
info@sdn.nl
redactie@sdn.nl

Adverteren:
Informatie over adverteren en de tarieven kunt u opvragen bij Kelly Verschoor
kelly.verschoor@sdn.nl

©2023 Alle rechten voorbehouden. Niets uit deze uitgave mag worden overgenomen op welke wijze dan ook zonder voorafgaande schriftelijke toestemming van SDN. Tenzij anders vermeld zijn artikelen op persoonlijke titel geschreven en verwoorden zij dus niet noodzakelijkerwijs de mening van het bestuur en/of de redactie. Alle in dit magazine genoemde handelsmerken zijn het eigendom van hun respectievelijke eigenaren.

Beste SDN-er,

We zitten nog na te genieten van de afgelopen Futuretech. Het was leuk om jullie allemaal weer te zien en te spreken. Naast een gave keynote van een echte expert uit het veld van AI, Tim Huckaby (over ethiek en AI), waren er gave sessies over Maui, Q# en Security. Voor elke developer wat wils en er was ruim genoeg tijd om met je peers of sprekers in discussie te gaan. Een mooie sfeerimpressie vind je elders in dit magazine.

Met de SDN Cast waren wij natuurlijk ook weer aanwezig en we hebben daar leuke gesprekken gehad met bijzondere mensen. Kijk ze nog eens na via sdncast.nl.

Nu de dagen weer langer worden en het aantal zonne-uren gaat stijgen, is niks fijner dan met een SDN-magazine in de ene hand en een verfrissend drankje in de andere hand lekker te genieten in je tuin of balkon. Het eerste hebben wij, samen met onze redactie en auteurs, voor je geregeld, dit SDN-Magazine is weer goed gevuld met interessante artikelen.

En ze zijn allemaal door echte mensen geschreven. In deze tijd waarin de AI-tools over elkaar heen buitelen om allerlei content te genereren, gaat er niets boven handwerk van de echte collega's uit het veld. Zij hebben wederom allemaal hun uiterste best gedaan om een mooi en interessant verhaal voor jullie op papier te zetten. Ervaringen uit het veld of grappige hobby projectjes.

Technologie gaat snel en in dit magazine zit een artikel om je te helpen je dag zo in te delen dat kennis opdoen daar een standaard onderdeel van is. Uiteraard hebben we ook artikelen over AI en ChatGPT, naast een ethische kant is AI ook bevooroordeeld en hoeft het allemaal niet zo eng te zijn. De auteurs leggen het graag uit. Zoek je nog een hobby projectje voor de overige vrije tijd, dan kun je eens kijken naar het artikel over Games en C#. Cloud is geweldig, maar alleen als je er goed mee omgaat en dat wordt iets dieper uitgelegd in het artikel over schalen. Devops is inmiddels wel redelijk bekend qua tools, maar het heeft ook nog een andere kant. Devs naar Devops gaat daar iets verder op in. En dan een interview met Dennis van NextGen, altijd leuk om ervaringen van een collega developer te lezen.

Kortom genoeg te lezen! Richard, Dennis, Meara, Jasper, Carlo, Frank en Twan, dank je wel! Redactie commisie fijn dat jullie deze mooie verhalen gevonden hebben!

Veel leesplezier en tot na de zomer!
Marcel





SELMA HELPS 1.3 MILLION PEOPLE MAKE ENDS MEET

How many people do you want to help?

Selma Kubat, IT project manager at Netcompany, helped to build the system that pays out national and social pensions to over 1.3 million people.

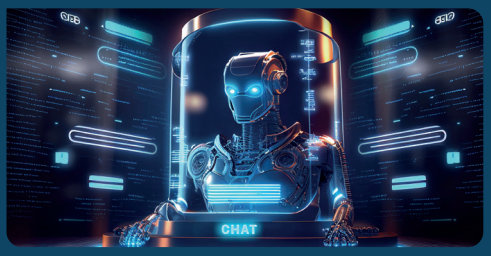
Do you want to work on IT projects that make a real difference to society?

Take a look at our open roles:

netcompany

In dit nummer

6 Will ChatGPT Replace Developers
Jasper Siegmund

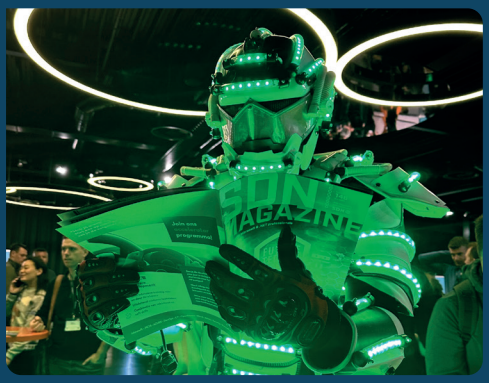


9 5 manieren om bij te blijven als software developer
Richard van Zandwijk

12 Bias in AI creeps in faster than you might think
Meara Leentvaar

15 Hacking video games with C#
Frank van der Stam

20 Future Tech Terugblik SDN



22 Best Practices voor .NETontwikkelaars in een cloud-gedreven wereld
Dennis Fatehmahomed

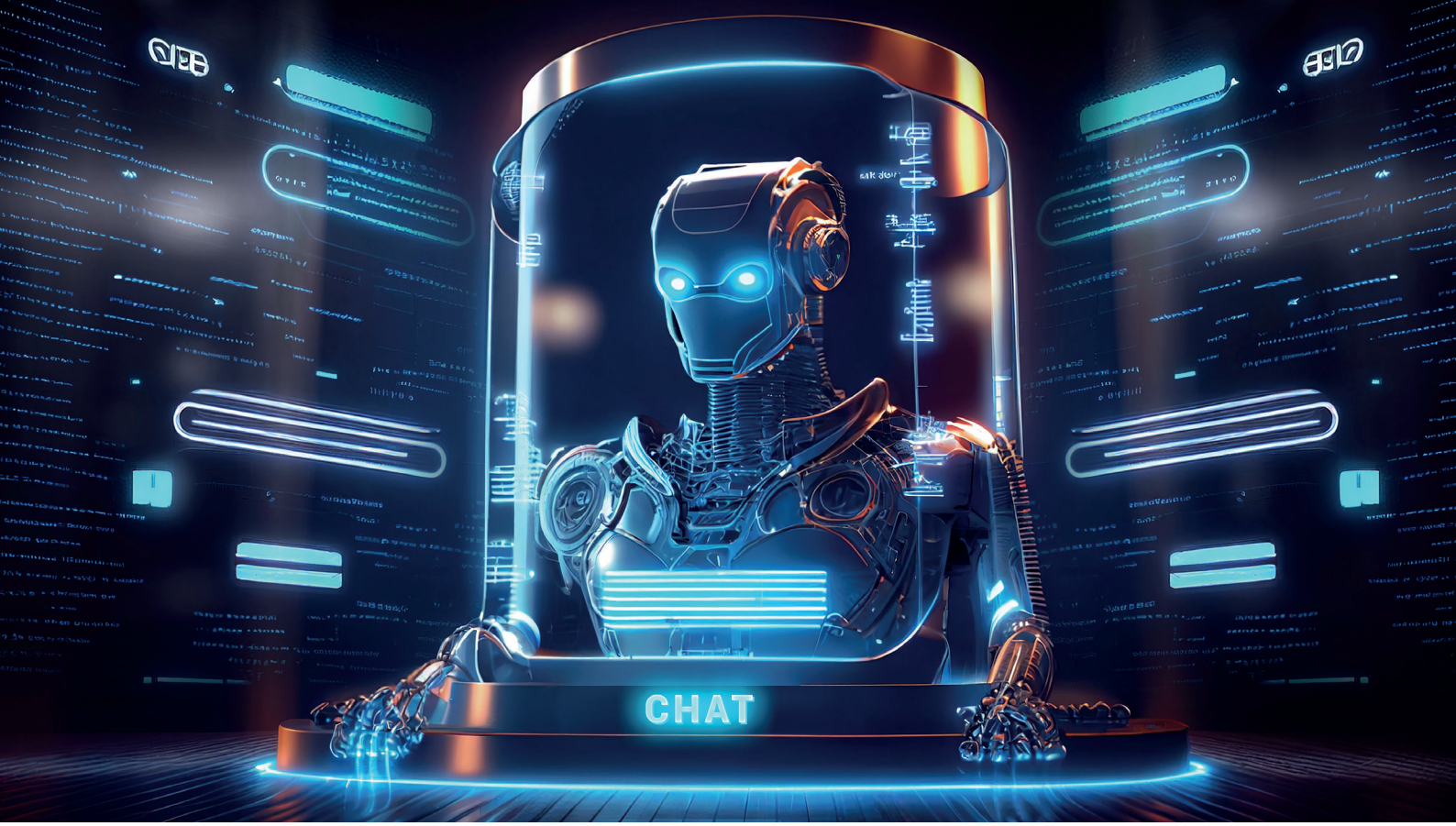
24 Van Dev naar DevOps
Carlo Kuip

28 Het schalen van Azure App Service
Twan Koot

6 Let's start with a short recap of what ChatGPT is. Built by OpenAI, which is backed by Microsoft, it's a model which in itself is the output of the GPT-3.5 algorithm (GPT-4 as of March 2023). Put in layman's terms; these models are the result complex algorithms which (kind of) resemble the neural networks in our human brains and are thereby able to perform 'intelligent' things.

9 We weten allemaal hoe het is. Overal om ons heen zien we nieuwe programmeertalen, nieuwe frameworks en libraries opduiken, die allemaal op een nieuwe en spannende manier een probleem proberen op te lossen dat veel mensen irriteert. Goede ontwikkelingen zijn dat!

15 Like many of us in the tech industry, I have enjoyed playing games from a young age. Sometimes I would want to do things in games that you were not really supposed to do. I would use open source tools and trainers to do this, but was clueless as to how any of those tools could possibly work.



Will ChatGPT **replace developers?**

Auteur: Jasper Siegmund

Artificial Intelligence models

Let's start with a short recap of what ChatGPT is. Built by OpenAI, which is backed by Microsoft, it's a model which in itself is the output of the **GPT-3.5 algorithm** (GPT-4 as of March 2023). Put in layman's terms; these models are the result complex algorithms which (kind of) resemble the neural networks in our human brains and are thereby able to perform 'intelligent' things.

Until ChatGPT, examples of these would be things like face recognition, insect identification, song discovery, self driving cars and more. With the exception of self-driving, the mentioned applications of AI are more or less in the recognition realm. Those models are "taught" how to recognize by supplying them with huge datasets of earmarked data, which it uses to find out the differences between, for instance, cats and dogs. Once trained, the model is then able to take a picture of a given pet and say whether that's a cat or a dog.

We're now starting to see a new breed of models which are more "creative". Instead of pure recognition, these rely on prompts to produce content as a reply. They're known as **generative** algorithms. ChatGPT is the main subject for this article, but a few months earlier DALL-E 2 caused a few stirs as well, with its ability to generate very impressive images based on textual prompts. And more in our own line of business, GitHubs **Copilot** (launched June, 2022) can actually help you write code.

Microsoft has quite aggressively started to include AI in its own products. **Bing** is probably what attracted the most media attention, when Microsoft openly took on Google to include these models as part of search and at first sight did a better job (on hindsight, it might not have been that great: https://edition.cnn.com/2023/02/14/tech/microsoft-bing-ai-errors/index.html).

So now that these models have begun generating content and literally: generating code, how will this impact the software development business?

What does ChatGPT do?

As you can see in the video below, ChatGPT is quite capable of writing code on its own. It's arguably better at it than some humans are. So how does it even know all this? Simply said: it relies heavily on the knowledge gathered on the Internet. Simply put: it has finishing reading the Internet, at least the content up to 2021 (for now). And it's quite capable of searching all of that content, picking out exactly the things that are needed to provide an answer to the given question and present that in a very readable and understandable way. That last part is important; it's more focussed on producing readable and understandable content than content that is factually correct. That said, it often comes up with the right answer, but not always. Nick Chapsas shows the ability of ChatGPT to write software.

Those of you who are old enough might still remember Ask Jeeves (which is now ask.com), a popular search engine before Google took over. Ask Jeeves was pretty much similar to what Google does, but featured a butler on it's front page which you could then "ask questions". The problem, as with Google, is that the "answers" are not really answers but hyperlinks to pages which might contain the answer you were looking for. This is the fundamental difference with ChatGPT: it really answers. And that answer is usually correct. Usually.

BIO

Jasper Siegmund
Cloud Solution Architect at mStack
Breda, Netherlands

Jasper works as an Azure specialized solution architect for mStack. What he enjoys most are somewhat complicated projects that bring a lot of value with them. Azure is his cloud platform of choice, .NET his toolset. He's also a frequent blogger, speaker and allround geek. And when not behind a laptop, he enjoys the outdoor life and aquascaping.



What doesn't ChatGPT do?

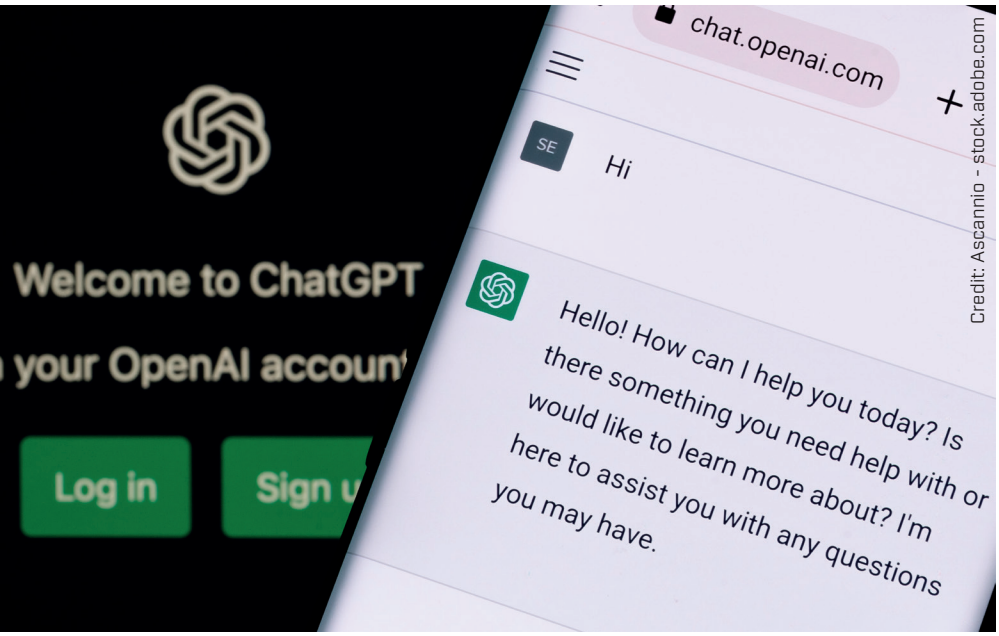
Given that ChatGPT is trained on "the entire internet" up to 2021, it by default lacks information from the last year. You can be sure that this will be fixed in future versions, but right now this still holds true. That means that when you ask it to give you the code to interact with some SDK, it might not know about the most recent versions of that SDK. And thus it might forget to tell you about that great new feature or recently discovered security problem. Also, "the entire internet" isn't always right. There's a lot of developers who use Stack Overflow in their daily routine. These developers will know that a lot of questions have multiple answers which are not equally as good. Fine-tuning of the dataset could of course easily fix that; for instance by only using the accepted answer or at least preferring that answer over others. But how will it know that the documentation of a given NuGet package is factually inaccurate or outdated because the developer didn't care about documentation that much? Lastly, there's only so much information that one can put into a single prompt. ChatGPT has contextual knowledge which serves as a "memory", so you can keep building on earlier prompts. But it won't yet build an entire solution for you, nor haven't I seen it commit things into git. We're definitely not at a point where you can give it some functional requirements and then have it build your app for you. Will we get to that point? I'm not sure.

It's all about the prompt

When DALL-E launched, marketplaces for prompts (yes, only prompts) started popping up. You could say that a large part of the creativity in the output is actually due to creativity in the prompt. A similar thing applies to ChatGPT: the better questions you ask, the better answers it provides. The above video by Nick Chapsas is an illustration of this, since Nick for instance quite specifically asks to provide a version of the code without heap allocations. There's a lot of developers out there who don't even know what heap allocations are or how these might impact performance. Once could reason that this tool might be more powerful in the hands of someone who knows what they're doing.

GitHub Copilot

We already briefly mentioned GitHub Copilot. For those who are not aware: Copilot integrates into your IDE to write code for you, based on what you're typing. So instead of asking it questions like with ChatGPT, it will proactively guess what you're doing and then try to write the code for you. And as you might expect, Nick Chapsas has a video on that as well. Nick Chapsas demos GitHub Copilot for C#



It's based on GPT-3, so a slightly earlier version of the same model which of course will be updated continuously. It will start providing suggestions for code lines or even complete functions, which you can easily accept so that you won't have to type it out yourself. I've used this in my day-to-day coding and it will definitely increase productivity. If you haven't tried it yet, do so!

So how will this impact software development?

Ok, having seen all this you might get a little worried. Has the time come where AI will start taking over your job as a developer? In my opinion, there's really no need to worry. Yet. ChatGPT is still in it's early days and since the moment it started to get busy, the availability has become an issue. There's a paid subscription now where you pay $20 monthly for unlimited use, which depending on how you use it may or may not be worth it. In it's current form, I regard it as a very useful tool which you can quite easily incorporate into your daily routines in order to improve your efficiency and therefore output. In essence: it can help you get thing done.

In the long term though, things might be different. Of course this technology will improve and these models will get better and better at writing code. So it's not that unlikely that at some point technology will start replacing the very inefficient developers by ones who maximize their efficiency by using tools like this. Not an immediate problem in a growing market, but recent tech lay-offs prove that our market doesn't always grow.

From a company or CxO perspective, your software development teams or department should at least be aware of this movement. When your developers can and want to incorporate these new tools in their daily routines, great: facilitate them! It'll most likely increase their output. If none of your developers seem to care, you're efficiency in the long term might not be what you want it to be. Chances are that they're ignoring other productivity boosters as well, being stuck in a specific way of working.

If ChatGPT is like a power tool, developers are like carpenters. Power tools didn't make carpenters obsolete, they did change their jobs. These tools allowed them to be more efficient, building the same house with fewer people or in less time. And the ones who master their tools have more relevancy than the ones who don't. So should you consider ChatGPT a threat or an opportunity? In my opinion it's both at this time, and you get to choose what you want it to be. Ignore it and it might become a threat to your job in the future. Use it and it might be one of your favorite power tools. But use it with caution, power tools can be dangerous.

Update March 15, 2023

I quickly wanted to revisit this article based on the newly released GPT-4 version. As you would expect, GPT-4 is an improvement over GPT-3.5. It's main improvements include: reliability, creativity, and handling of nuanced instructions. Of course we were not the only ones interested. The above mentioned Nick Chapsas released this video showing the model in action:

Impressive, right? It's even better than what we already had. It will bring a bigger increase to productivity. Next to this announcement and release, we also learned that Bings chat functionality is also based on the GPT-4 model (https://blogs.bing.com/search/march_2023/Confirmed-the-new-Bing-runs-on-OpenAI%E2%80%99s-GPT-4). So it's safe to say that this model wil rapidly outdate it's 3.5 predecessor and take over as the primary version. It will be interesting to see how quickly OpenAI will then be able to iterate to version 4.5 or 5.

So what does that mean for the conclusion of this blog? Nothing much, really. We already pointed out the enormous potential of the technology. We also stated that in time it might start to threaten jobs. Did that time arrive already? Only time itself will tell, but this might well be when it all started.



5 manieren om bij te blijven als Software Ontwikkelaar

Auteur: Richard van Zandwijk

We weten allemaal hoe het is. Overal om ons heen zien we nieuwe programmeertalen, nieuwe frameworks en libraries opduiken, die allemaal op een nieuwe en spannende manier een probleem proberen op te lossen dat veel mensen irriteert. Goede ontwikkelingen zijn dat! Maar helaas kunnen deze snelle ontwikkelingen je ook het gevoel geven dat wanneer je niet elk jaar een nieuwe taal of een paar frameworks leert, je achteropraakt op ontwikkelingen binnen de markt en jouw organisatie. Hoe houd je overzicht en blijf je up-to-date met deze grote set aan mogelijkheden? Hoe weet je dat je de juiste keuze maakt wanneer het aanbod zo groot is?

Wat?

Hou je kennis over je huidige framework up-to-date.
Wanneer er een aankondiging is van een nieuwe versie van Java, C#/.NET, ASP, Kafka, lees die dan! Dit kan een korte samenvatting zijn (in tekst of op YouTube, er zijn er talloze) of je kunt de changelog doorlezen, dat is aan jou. Maar blader er in ieder geval doorheen om te bepalen of er iets interessants tussen zit dat je tijd en moeite waard is.

Hou je vakgebied bij.
Of dit nou IoT, Big data in AWS of platformonafhankelijke mobiele ontwikkeling is, het is belangrijk om op de hoogte te blijven van de specifieke producten die je gebruikt - maar houd er rekening mee dat er meer in de wereld is dan alleen dat. Dit voelt als veel om te verwerken, maar daar komen we op terug in het "hoe?" gedeelte.

Probeer af en toe iets nieuws.
Zoals gezegd, zo nu en dan is er iets nieuws dat de wereld volledig zou moeten veranderen. Maar vaker wel dan niet heeft het toch niet de impact die we zouden willen. Soms komen er gelukkig ook technieken langs die blijven hangen in de praktijk in plaats van alleen als concept, en dan is het zeker de moeite waard om er eens naar te kijken.

Hoe?

Er zijn veel verschillende manieren om bij te blijven met nieuwe ontwikkelingen. Het doel is om het overzichtelijk te houden en je niet te laten overspoelen. Dit zijn de manieren die voor mij werken:

RSS-feeds. Zo simpel is het.
> Het is heel eenvoudig, maar het is voor mij een van de makkelijk-

BIO

Richard van Zandwijk

Richard is een ervaren software ontwikkelaar met een passie voor procesverbetering en -automatisering. Hij krijgt energie van het coachen en vooruit helpen van teams en het stroomlijnen van het proces om de business en development dichter bij elkaar te brengen. In het verlengde hiervan zet hij zich in als Tribe Lead bij Team Rockstars IT om de Microsoft community bij elkaar te brengen met het organiseren van events, in contact brengen van developers en het up-to-date houden van het kennisniveau.





ste manieren om informatie van verschillende websites te krijgen. Het is makkelijk om updates van frameworks en blogs van mensen die je interessant vindt te filteren tot het absolute minimum van waarover je wilt lezen. Algemene nieuwssites in ons vakgebied zijn bijna altijd te breed of te specifiek voor jou persoonlijk.

> Het maakt het bijzonder eenvoudig om framework updates bij te houden. Elk framework en elke library heeft een blog, maar je wilt hier niet op willekeurige momenten naartoe moeten navigeren om te kijken voor updates. Daarnaast wil je ook niet elke keer wanneer er iets gebeurt e-mails ontvangen (we krijgen al genoeg spam) wat weer voor ruis zorgt.

> Ik zou altijd gaan voor een RSS-reader zonder notificaties. Een variant met notificaties kan verleidelijk zijn, maar een belangrijk onderdeel van dit proces is dat je het in je eigen tijd doet en niet wordt overspoeld met informatie. Je leest wanneer je dat zelf wilt.

> De beste RSS-reader is eenvoudig, maar ik heb wel een aantal eisen. Zo wil ik mijn lijstjes kunnen delen tussen desktop en mobiel, omdat ik altijd wil kunnen lezen wanneer het mij uitkomt. Het liefste gratis natuurlijk. Persoonlijk gebruik ik Feedly, maar er zijn nog veel meer opties.

Schrijf code. Breng het in de praktijk.

> De leukste manier om een nieuwe taal of een nieuw framework te leren, is door het gewoon in de praktijk te brengen en er iets mee te schrijven. Wil je weten wanneer een specifieke pagina van een website eindelijk live gaat? Schrijf een Azure Container App om dat elke dag te checken. Wil je een to-do-app die je lijst precies naar wens beheert? Gebruik .NET MAUI of React Native om er één te schrijven.

Conferenties. Een dag van focus.

> Als je baan het toelaat, of als je een eigen bedrijf hebt, zou ik je ten zeerste aanraden om de investering in tijd en geld te doen om naar conferenties te gaan. Alles wat hot en happening is, zou je hier moeten vinden. Een groot voordeel is, is dat je een dag zonder afleiding hebt.

> Het is een geweldige manier om te netwerken in je vakgebied. Vind andere bedrijven, kijk wat je con-

Hoe houd je overzicht en blijf je up-to-date?

currenten doen en kom in contact met de mensen achter de tools die je gebruikt.

> De kwaliteit van de talks hier moet hoog zijn, met de mogelijkheid om later vragen te stellen of met de spreker te praten - iets dat moeilijker is bij het bekijken van een YouTube-video.

> Natuurlijk, als je meer in een niche werkt, kijk dan of de conferentie wel past bij je opdracht en doelen.

> Vergeet niet wat aantekeningen te maken terwijl je hier bent. Hoewel het zeker mogelijk is dat je er nooit meer naar zult kijken, werkt het dubbel verwerken van de informatie die je ontvangt om het beter te onthouden.

> Je kunt genoeg pennen, stressballen, mokken en waarschijnlijk T-shirts krijgen om jaren mee te doen!

Meetups. Maak kennis met de community.

> Ze zijn gratis!

> Meestal zijn er ook meetups bij jou in de buurt.

> Het is een geweldige manier om mensen in je vakgebied te ontmoeten, in een veel kleinere setting dan op een conferentie.

Schrijven. Deel je kennis.
Het lijkt misschien gek om dit hier in een artikel te adviseren, maar het delen van je kennis is een goede manier om kennis op te bouwen. Het dwingt je om je denkproces te heroverwegen en te valideren, met als groot pluspunt dat je iets teruggeeft aan de gemeenschap waar je ook van profiteert. Bij het maken van bepaalde ontwerpkeuzes in je werk, of na het onderzoeken van de beste database-engine voor jouw situatie, deel dan dat proces dan met de wereld.

De wereld van softwareontwikkeling is gebouwd op dit soort ervaringen, en het is precies waar mensen naar op zoek zijn wanneer ze met dezelfde dilemma's te maken krijgen.

Wanneer?

Een grote vraag voor mezelf is wanneer ik dit allemaal moet doen. Het klinkt natuurlijk mooi dat we de tijd moeten nemen om onszelf verder te ontwikkelen, maar een dag heeft maar 24 uur en ik moet ook nog slapen!

> Ik kan overdag niet elk bericht of elke update lezen, daar zou ik gek van worden. Ik probeer deze informatiestromen zoveel mogelijk te bundelen. Iedereen heeft zijn eigen proces, maar dit is een methode die persoonlijk heel goed bij mij past.

> Doordeweeks heb ik mijn RSS-feed die ik doorlees. Artikelen die de moeite waard zijn om later te lezen bewaar ik meestal voor als ik wat meer vrije tijd heb. Sommige zijn echter zo interessant dat ik ze gewoon meteen moet lezen.

> Ik probeer minstens één of twee conferenties per jaar bij te wonen.

Mijn proces in het kort:

> Pin om de dag artikelen vast die ik wil lezen.
> Lees de artikelen die interessant genoeg zijn bij voorkeur één keer per week.
> Ga naar minstens één grote conferentie per jaar.
> Reserveer af en toe wat tijd om een nieuw framework/taal uit te proberen.
> Solliciteer voor de meest interessante opdrachten/vacatures!

Het trekt me uit mijn werk en in de wereld van programmeren van vandaag.

> Als ik wat vrije tijd heb of op de bank zit en niks te doen heb (wie hou ik voor de gek, dit gebeurt nooit...), lees ik de artikelen die ik eerder heb opgeslagen. Ik heb een lopend document waarin ik aantekeningen maak van al deze artikelen, gewoon om ze nog eens door mijn hoofd te laten gaan. Daarnaast houd ik een lijst bij van dingen die ik verder wil onderzoeken, of misschien een project van wil maken - maar wanneer ik tijd heb om dat op te pakken, is weer een heel ander verhaal.

> Ik heb ook een regel om niet te veel tijd in werk te steken. Op dit moment werk ik in detachering, wat inhoudt dat ik 40 uur per week declarabel wil zijn. Dat betekent ook dat persoonlijke ontwikkeling niet zomaar gedaan is in die 40 uur. Van de zonnige kant bekeken: ik kan wel schakelen tussen opdrachten als het saai wordt en de opdrachten kiezen bij een bedrijf dat de technologie gebruikt waar ik beter in wil worden.

Key Takeaways

Maak je niet te druk. De RSS-feed geeft je de vrijheid om te kiezen hoeveel je wilt lezen en dit te doen wanneer het je uitkomt. De conferenties geven je de vrijheid om het vergroten van je kennis te combineren met leuke gesprekken en volledig te focussen. De meetups geven je de flexibiliteit om niet een hele dag/meerdere dagen op te geven, maar alleen een avond wanneer dat jou beter uitkomt. Experimenteer met wat voor jou persoonlijk werkt. Een deel van het plezier in ons werk is dat ons werkveld zo snel verandert dat we blijven evolueren en nieuwe dingen blijven leren. Laten we het leuk houden!

Bias in AI creeps in faster than you might think.

Auteur: Meara Leentvaar



The future is AI.

Artificial intelligence (AI) is revolutionizing the way we live and work, from powering virtual assistants and self-driving cars to aiding medical diagnoses and detecting fraud. With the rising popularity of what AI can do for you there has been a surge of new tools like ChatGPT or AI Builder within the Microsoft Power Platform. According to ChatGPT itself: the future of AI seems promising, but it is important that we develop and apply this technology responsibly and ethically.

What is bias in AI?

AI algorithms are trained using vast amounts of data, which means they can identify patterns and make predictions based on that data. However, if the data used to train the algorithm is biased, the algorithm may learn and replicate that bias in its decisions. A study published in 2021 found that language models, including GPT-3 (which ChatGPT has been trained on), have biases towards gender and race. The study found that the language models produced language that reinforced stereotypes, such as associating certain occupations with a particular gender or race.[1]

More recently ChatGPT has also been found to have a bias towards left-wing political parties in the Netherlands. When questions were asked regarding the 'stemwijzer' ChatGPT favored left-wing parties such as SP, DENK and D66. This bias is mainly due to the scientific and academic papers, which are generally more likely to be left-wing, ChatGPT has been trained on.[2]

Another example of bias in AI comes from a study published in 2020 found that AI-powered healthcare tools can exhibit racial bias. The study found that a popular algorithm used to determine who needs extra healthcare support was less likely to flag black patients as high-risk, even when they had similar health conditions as white patients.[3]

Does the problem lie within datasets?

Joy Buolamwani of the Netflix documentary Coded Bias, discovered that the datasets used to train AI may not be inclusive. While developing a smart mirror at MIT, Joy attempted to use facial recognition but found that her face was not recognized until she put on a white mask. The AI had been trained to recognize white faces much better than people of color. Creating new datasets with more variation is difficult, as it requires people to give up their

anonymity in public by contributing their faces to an AI database. However, biased data can be identified and removed through techniques like debiasing and reweighting. Examining datasets for bias and correcting problematic data is crucial to ensuring AI is inclusive.[4]

BIO

Meara Leentvaar

My name is Meara Leentvaar (She/her), I am a 27 year old Power Platform Developer/Designer at Ordina. I have a passion for IT & inclusivity and have done a minor in how bias in AI presents itself. Here are some trivial facts about me: I love to play games (Switch, PS4, PC, you name it), read books and manga. I consider myself a plant and a bunny mom which in all honesty isn't the best combination. I also have an unfounded fear of ostriches.



Double check and check again

Auditing AI systems is crucial to ensure they operate in a fair and ethical manner when making important decisions that can impact individuals and society. The process involves evaluating data, algorithms, and outputs to determine if they align with ethical and legal standards. Auditing helps identify potential biases, errors, or ethical concerns and enables stakeholders to take action to address these issues. It can also provide evidence in legal proceedings and improve the system. In summary, auditing AI systems is essential to ensure that they are transparent, accountable, and responsible in their decision-making processes.[5]

Diversity in AI

Fostering diversity in AI development involves ensuring a diverse team, using inclusive data sets, conducting bias audits, providing education on diversity, partnering with diverse organizations, encouraging diversity in AI research, and increasing transparency. This is essential to prevent AI systems from perpetuating biases and stereotypes, and to ensure ethical and inclusive AI. Microsoft is one of the companies that advocates for diversity within AI with their responsible AI practices.[6]

Ethical thinking is crucial thinking

To create ethical standards in AI, it is necessary to identify relevant ethical principles such as fairness, transparency, accountability, privacy, and autonomy. Guidelines should be developed based on these principles and a broad range of stakeholders should be involved in the process. A review process should be established to ensure the guidelines are updated as needed, and education and training should be provided to developers and users. To encourage compliance, incentives and sanctions may be necessary, such as recognition programs, certification schemes, or regulatory mechanisms. By following these steps, it is possible to create ethical standards that promote responsible and trustworthy AI systems that benefit society while minimizing risks and harms.

Conclusion

AI has the potential to revolutionize various aspects of life, but as with any technology, there are potential risks and challenges. One of the most significant challenges facing AI is bias, which can be introduced into algorithms through biased datasets. The consequences of such biases can be far-reaching, perpetuating stereotypes and limiting opportunities for certain groups. It is, therefore, essential to develop and apply AI technology responsibly and ethically, taking steps to foster diversity in AI development, audit AI systems for bias, and establish ethical guidelines that promote fairness, transparency, accountability, privacy, and autonomy. These guidelines should be updated regularly, and education and training should be provided to developers and users. Through such efforts, it is possible to harness the power of AI in a way that benefits society while minimizing risks and harms. ●

Sources

[1]: Li Lucy and David Bamman. 2021. Gender and Representation Bias in GPT-3 Generated Stories. In Proceedings of the Third Workshop on Narrative Understanding, pages 48–55, Virtual. Association for Computational Linguistics. https://aclanthology.org/2021.nuse-1.5.pdf
[2]: ChatGPT has left-wing bias in Stemwijzer voting advice application. (2023, March 8). Leiden University. https://www.universiteitleiden.nl/en/news/2023/03/chatgpt-has-left-wing-bias-in-stemwijzer-voting-quiz
[3]: Ziad Obermeyer et al., Dissecting racial bias in an algorithm used to manage the health of populations.Science 366, 447-453(2019).DOI:10.1126/science.aax2342 https://www.science.org/doi/10.1126/science.aax2342
[4]: Joy Buolamwini: examining racial and gender bias in facial analysis software - Google Arts & Culture. (n.d.). Google Arts & Culture. https://artsandculture.google.com/story/joy-buolamwini-examining-racial-and-gender-bias-in-facial-analysis-software-barbican-centre/BQWBaNKAVWQPJg?hl=en
[5]: Auditing Artificial Intelligence. (2018) Isaca. https://ec.europa.eu/futurium/en/system/files/ged/auditing-artificial-intelligence.pdf
[6]: TechCrunch is part of the Yahoo family of brands. (2019, June 17). https://techcrunch.com/2019/06/17/the-future-of-diversity-and-inclusion-in-tech/?guccounter=1&guce_referrer=aHR0cHM6Ly9oYnlub3JnLw&guce_referrer_sig=AQAAAHyfm27AAoTUnBdQGULUsHrIoI5A8DU66tf2uL_PBkOvQ7rzvKto36Xkvbwyuv7jfNj7wk756mklxPLDZUiSZKQRC-mEMlb6NmcjlReNQgscNFeAiIAXePZI5GlTq0j5V78NdmHZBh4uEj8oneGra11mngutO8irH_DxNZwvGD7L

Azure Lowlands

TECH, MUSIC & FOOD
CONFERENCE-FESTIVAL

29 JUNE 2023

9:00 TILL 21:00

UTRECHT
DE FABRIQUE

Keynotes by

Sarah Young

Lars Klint

Bands

Back to Woodstock

The DOORS tribute

Sixties-Revolution

Get Tickets

azurelowlands.com

Hacking video games with C#

Like many of us in the tech industry, I have enjoyed playing games from a young age. Sometimes I would want to do things in games that you were not really supposed to do. I would use open source tools and trainers to do this, but was clueless as to how any of those tools could possibly work. One day I found a tool written in C# with some bugs. As a C# developer, reading through this tool felt less daunting, and it led me on a path to learn game hacking and reverse engineering. In this article I will show you how easy it can be to do a very basic attack on a game, and how simple and short the code can be.

Auteur: Frank van der Stam

Disclaimer: I mod offline games only. Cheating in online games is lame. Use the tricks I teach you in this article for offline games only. Don't use these tricks to get around paying for a license in programs.

The best way to learn is by doing. A remake of a popular video game "Resident Evil 4" has recently been released. In honor of this, I opted to target the original Resident Evil 4 from 2005 for this article. Let's modify that game so that we have infinite ammo. Let's talk strategy. The game will allocate some memory to store a variable for ammo (the number of bullets that remain in our gun). This number could be in a class instance, or a singleton, we don't know, but it will definitely be somewhere in memory. We can use a memory scanner to find this memory location. Once found, we can use a debugger to find out what parts of the game's code is reading or writing to and from this memory location. We can then use a disassembler to turn the game's machine code into readable assembly, from there we can decipher which parts of the code are decrementing the value each time we fire the gun. We can then replace that code with our own code that does nothing, effectively giving us infinite bullets! Sounds easy enough right? Let's dig in.

Memory scanning

When a native program is started on Windows, the executable file is copied into memory. This file contains the machine code (ones and zeros) that make up the game's code. Everything the program does will also need memory in one way or another. In our case, there will be a player object somewhere in memory. There will also be a level, some enemy objects, etc. All of these things will be allocated on a memory location called the "heap". The heap is dynamic, and everything will go in a different spot every time. Things you find on the may no longer be there after a certain interaction with the game, like changing levels, or restarting the game (which will clear all of the heap of course). We will be scanning through all of this memory, mainly through the heap. If a game says it is using 12gb of ram in task manager, we will be scanning through the largest portion of this 12gb of memory. Multiple tools exist to assist with memory scanning, I will be using

BIO

Frank van der Stam

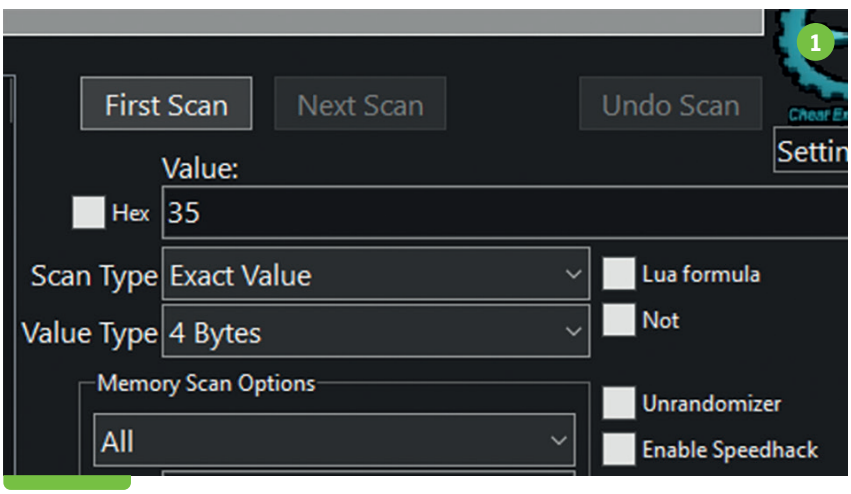
I am a 27 year old .NET professional that likes to mess around with low-level code in my spare time, from understanding hardware with microcontrollers to reverse engineering video games with hacking tools.



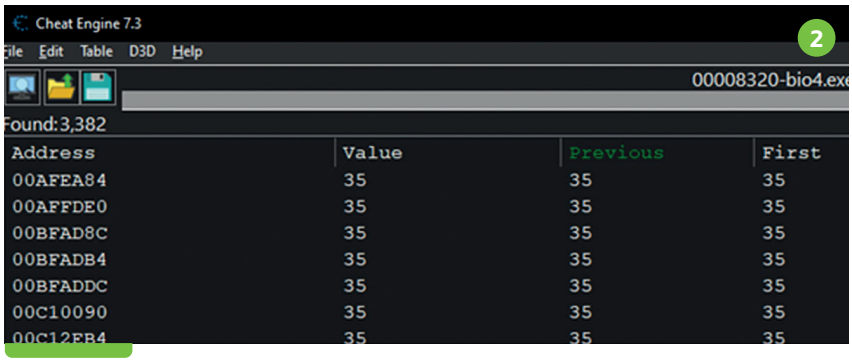
Cheat Engine for this. Cheat Engine is a Swiss knife when it comes to reverse engineering games.
The first step is to reason about, or do an educated guess, as to what data type will be used to store the memory value that you want to find. In our case, it would not make sense for the ammo to be stored in a floating point data type, or in a string. Most likely it would be stored in a 32 bit integer.

Let's modify that game so that we have infinite ammo.

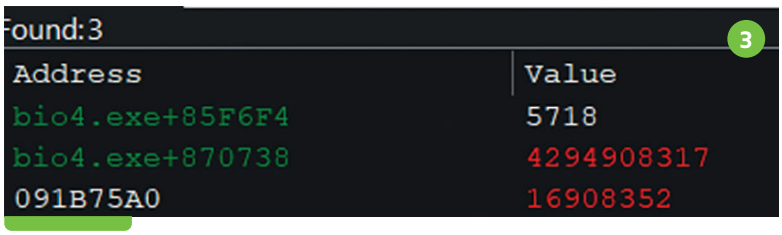
The second step is the filtering strategy. In our case, we know the value of the ammo. Sometimes, you're looking for an internal game engine value, like the ID of the current map or the current soundtrack. In those cases, you don't know the value.
Besides knowing the value, it is important to be able to influence the value, or to at least to know when it has changed. The reason for this is that we will go through all memory associated with the program, and do a little elimination game.
We might start off with an "unknown initial value", in which case we start with all memory, or we have a known initial value, in which case we already start filtering out memory locations with values that do not meet the criteria of our initial value.
After the initial scan, we will do another scan, to do another round of elimination. This time, we don't scan all memory again, instead we use the locations that we found in our previous scan. Each scan makes the list of possible memory locations smaller.
We can influence the ammo value by shooting in-game, right before we do a next scan, to change the value. We then either scan with an exact value, where we filter out any memory locations whose value is not equal to the ammo, or we scan for values that have decremented since last scan or we could scan for values that have simply "changed" in any way, increased or decreased.
Once we only have a handful of memory locations left, we can write to them one by one and check the game if it changes our ammo. You probably have at least 10 addresses most of the time that could potentially be the thing you are looking for. But that means the other 9 are for something else, and writing to those addresses could potentially crash the game.
You can also imagine that the UI code will have a copy of the ammo variable, and use that copy to update the UI, so that players can see it on their screen. This value will be overwritten by the real variable every frame. If we would overwrite this value, we might see the game's UI update to the new value, but as soon as we shoot the gun, the value gets brought down to the original, minus one. This is why we write to each variable, one at a time, and then fiddle with the game to see if it had the correct effect.
Being skilled at manual memory management in C/C++ makes it easier to understand how memory works, but will also make your educated guesses better. It will also help you understand what interactions with the game would be helpful while scanning.
Afbeelding 1
Afbeelding 2
As you can see, I tried the elimination game with exact values. I looked at the game's UI to determine what value to scan for. Unfortunately, this approach ultimately led me to eliminate all memory locations.
On the 2nd attempt, I started scanning for an "unknown initial value", and scanned for a "decremented value" each consecutive scan after that, shooting the gun in-game in between each scan. This ultimately boiled the list down to 3 memory locations. At first sight, the values of those locations didn't make sense at all for them to hold the ammo. **Afbeelding 3**



(picking a data type and a value for the first memory scan in cheat engine)



(scan results)

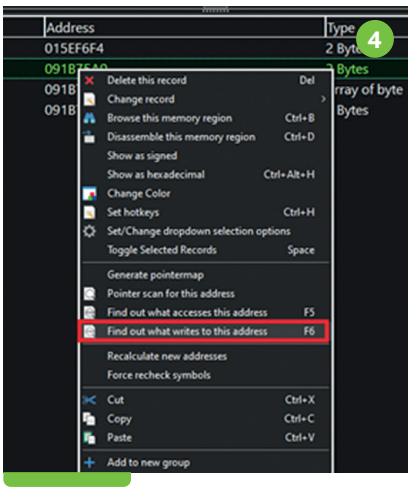


(3 resulting memory locations)

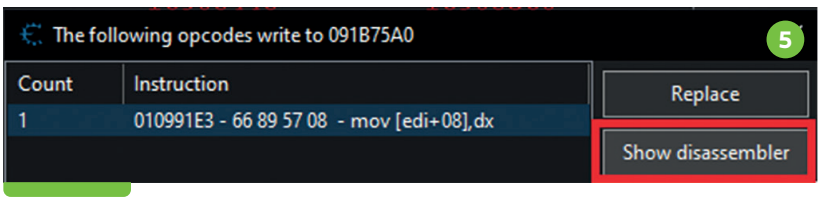
These values seem far too high - I do not have over 5000 bullets.
After watching these values carefully while interacting with the game, it started to become apparent to me that the last location (0x091B75A0) encodes the ammo in only the first 3 bytes. Besides that, each shot decrements the value by 8. Overwriting those first 3 bytes with 128 caused the ammo in-game to jump to 16. Bingo! We have found our magic memory location.

Defending against memory scanning

While memory scanning, we found a strange encoding of ammo, in increments of 8. From a "security" or "anti-cheat" perspective, this encoding was very trivial to defeat, but it is not always so easy. Some games will have a secondary memory value to check if the first value has been tampered with. This secondary value can be a simple copy of the original, but sometimes a complicated formula is used to make this second value differ from the first. That makes it hard to find in scans while you can still compare the two values with the same formula, to make sure neither has been tampered with.



(attaching a debugger)



(triggering a memory breakpoint)

A reasonable defense would be to have seemingly random floating point values, or even strings, in a linked list. Rather than decreasing the value for each gunshot, you move to the next element in the linked list; sometimes shooting would cause the current number to go up, sometimes to go down. You can then use a dictionary to map this seemingly random float/string to the actual amount of ammo, to update the UI or to check if the gun is empty. This can of course also be defeated, but it would take a significant amount of time to find and understand the code responsible for this.

Finding and disassembling the code

The next step is to find out what code is decrementing the number. We can ask Cheat Engine to show us what code is writing to our memory address. **Afbeelding 4**
This will attach a debugger to the game, with a memory breakpoint on our memory location. After setting this up, we have to shoot again in-game, to trigger some part of the game's code to decrement the value, and trigger our memory breakpoint. **Afbeelding 5**
We get another memory location out of this, and some assembly. This memory location is from inside the region of the game's executable. It's important to realize that this address is static: it will not change in between levels or restarting the game, unlike the ammo variable location. From here, clicking on "show disassembler" will have Cheat Engine disassemble the machine code into assembly for us: **Afbeelding 6**

Some games will have a secondary memory value to check if the first value has been tampered with.

These are the assembly instructions that the game is executing. Might look scary at first, but if you take your time, it isn't that hard to understand. The line where our breakpoint triggers is on a mov instruction. Mov moves a value from one place to the other. With that in mind, it becomes apparent that [edi+0x08] (which just boils down to a number, another memory location) contains the location where ammo is stored and the dx CPU register contains the newly decremented ammo value. Mov is simply copying the new value into the location where the old value is stored. In this case, that is really all we need to understand when looking at this code, we don't care about the other instructions. For now we can write down the address **bio4.exe+3091E3**.

Fun fact: there are reverse engineering communities that completely reverse engineer the assembly of games back into human readable code. A notable example of one such community centers around zelda games: https://zelda64.dev/.

The hackity part

Now that we have our disassembled code, and we know what instruction we want to overwrite, it is time to get started with some C#.
There are a few ways that we can access another process, like DLL injection. From C#, we will take the simple path for now, and let the win32 API handle this part for us. For the uninitiated, the win32 API is an API that Windows itself provides. Any native windows application uses this API. That includes the C# standard library, though most of its use is hidden behind common objects, or indirectly handled by the runtime. The functions we're after reside in kernel32.dll. We want a function called "WriteProcessMemory". This function accepts the handle of a process as a parameter, along with an address to write to, and an array of bytes to write to that address. As the name suggests, this function will write memory to the specified process.
We will need to do some P/invoking to ask Windows to execute this function for us, and we're going to abstract most of the required parameters away in an extension method: **Afbeelding 7**
Now that we can write memory, we need to get a reference to the game's running process (bio4.exe just happens to be the name of the executable). We need to replace the single 4 byte wide mov instruction with multiple nop (no operation, it does nothing) instructions, because nop is only 1 byte wide. The hexadecimal value of nop is 0x90. We can define an array of these, and write to 0x3091E3. We can't write directly to 0x3091E3 - this address is relative to the location where the executable is loaded in memory. We can use the MainModule's base address to calculate the absolute address to write to. **Afbeelding 8**
After running this program, I can shoot my gun infinitely, without my ammo ever decreasing!
This trick is not limited to changing game code, we can also overwrite memory values themselves. The game has an in-game shop where

```
bio4.exe+3091D7   8D 14 C5 F8FFFFFF   lea   edx,[eax*8-00000008]
bio4.exe+3091DE   83 E1 07            and   ecx,07
bio4.exe+3091E1   0B D1               or    edx,ecx
bio4.exe+3091E3   66 89 57 08         mov   [edi+08],dx
bio4.exe+3091E7   5E                  pop   esi
bio4.exe+3091E8   5F                  pop   edi
bio4.exe+3091E9   B0 01               mov   al,01
```

6

(disassembly in cheat engine)breakpoint)

```
public static class Kernel32
{
    //Win32 API method import to write memory by a process's handle
    [DllImport("kernel32.dll")]
    private static extern bool WriteProcessMemory(IntPtr hProcess, IntPtr lpBaseAddress, byte[] lpBuffer, uint nSize, out uint lpNumberOfBytesWritten);

    //Extension method
    public static void WriteMemory(this Process process, long address, byte[] buffer)
    {
        WriteProcessMemory(process.Handle, (IntPtr)address, buffer, (uint)buffer.Length, out _);
    }
}
```

7

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        //Get a reference to the game process
        var process = Process.GetProcessesByName("bio4").FirstOrDefault();

        //The address will be relative: we have to start from the base address of the main module. The main module is basically the .exe
        var address = process.MainModule.BaseAddress.ToInt64() + 0x3091E3;

        //0x90 is a NOP instruction -- "no operation"
        var bytes = new byte[](0x90, 0x90, 0x90, 0x90);

        //Overwrite the game's code
        process.WriteMemory(address, bytes);
    }
}
```

8

```
//Overwrite the game memory instead of code
address = process.MainModule.BaseAddress.ToInt64() + 0x85F7;
bytes = BitConverter.GetBytes(999999999);
process.WriteMemory(address, bytes);
```

9



10

With some guessing, trial and error, simple C# code and an hour or two of time, we can mess around

you can buy weapons and healing items for the "ptas" currency. We can scan for the value of ptas, and apply a similar trick. I ended up finding this address at offset 0x85F708. By just adding a couple more lines of code, we can give ourselves a lot of in-game money: **Afbeelding 9**

A conversion from a high integer value (high because we want lots of money) to bytes is needed here, BitConverter is perfect to deal with that problem – it will give us the binary representation of that number, which we can write back to the game's memory. The effect of running this is immediately apparent in the game: **Afbeelding 10**

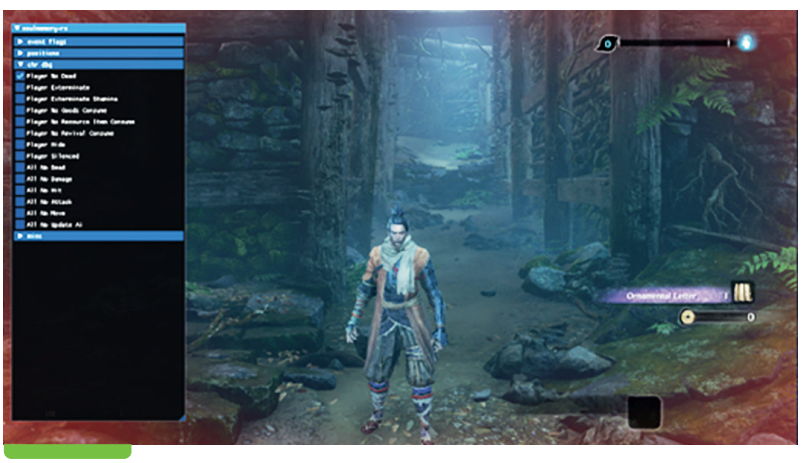
We can now spend to our hearts content!

Conclusion

With some educated guessing, trial and error, some very simple C# code and about an hour or two of time, we can mess around with games, give ourselves god-like abilities and get the "reverse engineer's dopamine rush".

Of course, these are cheap hacks to make the game very easy. It would be more interesting to create more meaningful modifications. To achieve that, we would have to use Ghidra and other tools to dig through the game's code and find out how it's systems work. For instance, most game engines use a proprietary format to store assets, like textures and sounds. By reverse engineering the code that loads these resources, we could make our own tools to read these proprietary formats, in order to alter them. This would allow us to change how the game looks, or load our own sounds.

Another possibility is function hooking. While technically possible from C#, this is easier from C/C++/Rust. Function hooking is when you load a chunk of code into a game process (usually via DLL injection). You then overwrite the first few bytes of a game function with a jump instruction, that jumps to your own code. You effectively "hijack" some of the games functionality. An example of this is hijacking a graphics related function, like a direct X function that renders the game. We could then, before rendering the game itself, use the hijacked graphics context to draw our own stuff on top of the game, before jumping back and letting the game render. This would allow use to draw our own menu/overlay inside the game! ●



(in-game GUI overlay hack in Sekiro: Shadows die twice)



Futuretech **terugblik**

We zitten nog na te genieten van de afgelopen Futuretech. Het was leuk om jullie allemaal weer te zien en te spreken. Naast een gave keynote van een echte expert uit het veld van AI, Tim Huckaby over de implicaties van de ethische vragen van AI, waren er gave sessies over Maui, Q# en Security. Op https://www.youtube.com/@futuretech2456 zullen de opgenomen sessies allemaal beschikbaar komen.

De SDNCast (https://www.youtube.com/@SDNCast of https://sdncast.nl) was natuurlijk ook weer aanwezig. We hadden het genoegen om een kort gesprek te hebben met Tim en Sammy over AI en over hun sessies. Daarna hebben we veel bezoekers gevraagd naar hun Future Tech, welke technologie is of zal voor jou heel belangrijk worden de komende tijd. Daar kwamen er verrassende antwoorden uit. Kijk ze nog eens rustig na.

Hou onze socials en magazine in de gaten voor Futuretech 2024! Dat wordt nog gezelliger, mooier en interessanter. Heb je in de tussentijd tips of tops, meldt dat dan via info@sdn.nl of redactie@sdn.nl



FUTURE TECH 2023

THE IT CONFERENCE FOR MICROSOFT TECHNOLOGIES





Best Practices voor .NET ontwikkelaars in een Cloud-gedreven wereld.

In de snel evoluerende wereld van technologie speelt de rol van .NET ontwikkelaars een cruciale rol bij het vormgeven van innovatieve oplossingen. Maar wat drijft deze professionals en hoe blijven ze voorop lopen in een steeds veranderend landschap? In dit interview gaan we in gesprek met Dennis Fatehmahomed, een ervaren .NET ontwikkelaar werkzaam bij The.NextGen, om zijn ervaringen, uitdagingen en inzichten te ontdekken. Duik met ons mee in zijn verhaal en ontdek hoe hij succesvol is geworden.

Hi Dennis, kan je beginnen met wat over jezelf te vertellen?
Ik ben Dennis Fatehmahomed, een 39-jarige .NET ontwikkelaar en solutions architect uit Rotterdam. Ik ben al meer dan 15 jaar werkzaam in dit vakgebied, waarvan de laatste vier jaar bij IT consultancy The.NextGen. Ik ben een fullstack developer (Frontend & Backend), mijn focus frontend is Angular en aan de Backend ben ik voornamelijk bezig met Cloud, technische architectuur, software development conform design patterns en Microsoft standaarden. Ik heb diverse rollen ingevuld: developer, lead developer en solution architect (en soms ook een combinatie van die 3).
Ik heb een passie voor het begrijpelijk maken van technologie voor mensen met of zonder technische kennis. Ik ging bij The.NextGen aan de slag om voor interessante bedrijven uitdagende opdrachten aan te gaan waar ik een grote impact kon hebben terwijl mijn persoonlijke ambities en groei ondersteund worden. Ik heb gewerkt voor bedrijven als CAK, Trust, het Ministerie van Defensie, Rabobank en ABN Amro.

Laten wij meteen met de deur in huis vallen, wat vind je het leukst en interessantst aan het werken als .NET developer?
Een van de meest interessante aspecten van mijn werk als .NET developer is dat nieuwe ontwikkelingen en innovaties onze industrie vormen. We zitten niet in een stagnerend, langzaam bewegend vakgebied, integendeel. Onze sector beweegt snel. Als .NET ontwikkelaar is het daarom van essentieel belang dat je de juiste balans vindt tussen je huidige taken en het op de hoogte blijven van recente ontwikkelingen. Dat houdt het interessant en verfrissend. Maar dan moet je wel moeite doen om op de hoogte te blijven van deze ontwikkelingen en daardoor heb ik het belang ontdekt van een aantal best practices: het schrijven van schone en onderhoudbare code, het naleven van industriestandaarden en coding standards, en het voortdurend leren en up-to-date blijven met de laatste trends en ontwikkelingen.

Wat is een van de ontwikkelingen geweest die een grote invloed gehad heeft op jouw werkzaamheden en wat was de impact?
Een van de meest impactvolle technologieën die het meeste invloed heeft gehad op mijn dagelijkse werkzaamheden is de grootschalige adoptie van de Cloud. De switch naar Cloud zal de toekomst van .NET development blijven bepalen, aangezien steeds meer bedrijven hun applicaties naar de Cloud verplaatsen als een manier om de efficiëntie en schaalbaarheid te vergroten. Of je nu werkt met AWS of Microsoft Azure, Cloud is overal om je heen en allesomvattend. En dat resulteerde in nieuwe uitdagingen en nieuwe standaarden.
Cloud heeft de snelheid waarmee dingen gedaan worden beïnvloed, alles gaat sneller waardoor wij als developers ook sneller moeten schakelen. Daarnaast heeft het de manier waarop .NET ontwikkelaars werken veranderd. Het introduceerde een stukje infrastructuurontwikkeling in .NET. En als iets niet naar behoren werkt, moet je uitzoeken wat er mis is. Is het de interface met de Cloud infrastructuur? Of een bug in de code zelf?
Bovendien leidde de adoptie van Cloud naar de verandering van één monoliet systeem dat alles doet, van bijvoorbeeld het verwerken van een bestelling tot het versturen van de



Dennis Fatehmahomed

factuur, naar in een veelvoud van kleinere systemen op de Cloud. Het ene kleine systeem verwerkt de bestelling, terwijl het andere de factuur verstuurt. Hierdoor wordt het makkelijker om het overzicht te verliezen en moeilijker om het gehele systeem te beschrijven, wat juist weer makkelijker is met een monoliet systeem.
Als laatste was er aanvankelijk scepsis over beveiliging van de Cloudtechnologie. Was het wel veilig genoeg? Natuurlijk is de beveiliging van Cloud platforms ontzettend scherp en hebben bedrijven ook hun security en compliance afdelingen opgeschaald om eventuele beveiligingsrisico's die gepaard gaan met de Cloud op te lossen. Waar ligt dan het meeste gevaar? In de kleine menselijke foutjes.
Menselijke foutjes blijven in het algemeen de oorzaak van de meeste beveiligingsrisico's, voornamelijk bij nieuwe systemen. Daar gebeurt het heel vaak. Het is dus extreem belangrijk om ervoor te zorgen dat je een structuur hebt om ervoor te zorgen dat de code aan alle eisen voldoet. Gelukkig zijn er allerlei tools die daarbij helpen tijdens het ontwikkelen, al helemaal voor de Cloud. Maar wat ook nodig is om deze menselijke fouten te beperken en om alle nieuwe elementen die door de Cloud bij .NET development zijn geïntroduceerd behapbaarder te maken, zijn goede en duidelijke standaard werkwijzen.

Wat voor standaard werkwijzen gebruik jij om ervoor te zorgen dat de menselijke foutjes beperkt blijven en alles gestructureerd verloopt?
Bij veel bedrijven waar ik met C# gewerkt heb, zorgden wij ervoor dat onze standaard werkwijzen op orde waren. Zo werk ik veel met de microsoft standard practices, juist omdat ze tegenwoordig erg duidelijk en secuur zijn. Ze bieden developers een houvast, een leidraad en een richting. Het geeft je ook de houvast om goed te motiveren waarom je een beslissing maakt en waarom je bijvoorbeeld op sommige punten afwijkt van de microsoft standard practices. Deze standaarden bieden zo een goed kader en geven teamleden een goed begrip van de redenen om voor bepaalde patronen te kiezen. Dit maakt het ook makkelijker om, als je aan een project werkt, een gestructureerd resultaat aan de klant kan geven aan het eind van het project waar zij zelf makkelijk aanpassingen in kunnen maken. Je moet ze dus niet volgen naar de letter, maar naar de geest. De balans hierin moet je zelf vinden en in de gaten houden.

Jij had het er eerder over dat je het interessant vindt dat er veel nieuwe ontwikkelingen zijn binnen je vakgebied, maar hoe blijf je op de hoogte van deze nieuwste ontwikkelingen en updates en waarom is dat dus zo belangrijk?
Als ontwikkelaar kan je de nieuwe ontwikkelingen, zoals wij gezien hebben met de Cloud, niet negeren. We moeten dan ook op de hoogte blijven van veranderingen in ons veld, deze omarmen en voortdurend blijven leren. Elke dag een half uurtje de laatste ontwikkelingen tot je nemen, bijvoorbeeld op medium, is iets wat elke developer zou moeten doen. Daarnaast is het belangrijk om goede afspraken met werkgevers te maken over het volgen van relevante cursussen en verdiepingen, of nieuwe certificaten te behalen. Des te meer kennis en certificaten je hebt, des te meer je ook weer waard bent op de markt. Door de standaardisatie van de Cloud en de standaard werkwijzen wordt het ook makkelijker om voor tijdelijke krachten, zoals detacheerders, een team te komen versterken aangezien ze zich snel aan de werkwijze van de opdrachtgever kunnen aanpassen. Als je dan meer waard bent, vertaalt dat zich weer direct naar je salaris. .NET developers moeten ook niet bang zijn om nieuwe opdrachten aan te pakken. Uit mijn ervaring heb ik gemerkt dat het je scherp houdt, aangezien je telkens voor nieuwe uitdagingen komt te staan en dat zorgt ervoor dat je kan blijven groeien. Via The.NextGen ben ik bij meerdere opdrachtgevers aan de slag gegaan waarbij ik zelf kon aangeven waar ik op zoek naar was en voor ik het wist was ik aan een nieuwe uitdaging begonnen. Het is altijd een leuke beproeving om obstakels te overwinnen en een oplossing te vinden die werkt voor de klant. De blootstelling aan verschillende klanten en systemen verbreedt mijn perspectief en creëert de mogelijkheid om mijn expertise en kennis met anderen te delen. Omdat de technologie zich blijft ontwikkelen en omdat er steeds meer vraag zal zijn naar ontwikkelaars die veilige, schaalbare en betrouwbare software kunnen maken, is het van belang voor ontwikkelaars om nieuwe trends te omarmen en voortdurend leren tot een onderdeel van hun routine maken om te slagen in het steeds veranderende veld van .NET development, zodat we betere en toekomstvast oplossingen blijven bedenken en leveren aan alle klanten, of het nu gaat om de hele groten of de kleintjes.

Van Dev **naar DevOps**

Auteur: Carlo Kuip



Door een collega werd ik gevraagd om een artikel te schrijven met DevOps als topic en daarin te reflecteren op mijn ervaringen. De afgelopen tien tot vijftien jaar heb ik aan een aantal organisaties richting gegeven en zelf ook deelgenomen in de verandering van Dev naar DevOps. Daarnaast ben ik in mijn huidige rol, waarbij Cloud migraties de kern van mijn werk vormen, ook betrokken bij het ontwikkelen en geven van trainingen over DevOps om engineers voor te bereiden op een andere werkwijze. Die achtergrond biedt een mooi bruggetje om eens terug te kijken en te zien of er een essentie van de verandering van Dev naar DevOps samen te vatten is.

Beperkingen

Het meest aansprekende aspect in deze verandering vind ik de wrijving die ontstaat bij de overgang van Dev naar DevOps. In mijn beleving ontstaat dat door het domein van bijna onbegrensde mogelijkheden (Dev) naar de beperkende realiteit van de operationele systemen (Ops) moet worden vertaald. Dat is een wereld waarin een andere vorm van engineering wordt gevraagd, gekoppeld aan een sterke discipline en hygiëne, welke bij softwareontwikkeling niet altijd even duidelijk zichtbaar is. Ook de kennis over platformen, netwerken en aspecten zoals observability zijn vak geheel nieuw of grote onbekenden. Dat brengt dus veel nieuwe uitdagingen met zich mee.

"Wrijving geeft warmte" zegt men wel eens om duidelijk te maken dat wrijving ook positief kan zijn maar het beschrijft ook zekere weerstand die overwonnen moet worden. Die spreekwoordelijke drempel die men over moet in deze verandering van Dev naar DevOps, is geen eenvoudige en verlangt dat je als ontwikkelaar goed beslagen ten ijs moet komen. Echter je hoeft als Dev ook niet bang te zijn. Immers, je capaciteit om te leren en analyseren is al onderdeel van je dagelijks werk. Dat is een ijzersterke basis. Er is nog wel een aanvullende voorwaarde en dat is doorgaande steun vanuit het management/leiderschap van je organisatie. Zonder die steun is de kans op succes bij het afleggen van deze reis nihil.

Kennis

Elk domein verlangt zijn specifieke kennis om daarin succesvol te kunnen opereren, het is niet anders met DevOps. Naast kennis hebben zul je ook een aantal vaardigheden moeten ontwikkelen en beheersen om operationeel aan de slag te kunnen en productief te zijn.
Een stuk training kan met name op het gebied van te volgen werkwijze en aan aantal praktische zaken een goede invulling zijn maar er zijn ook meer theoretische aspecten en achtergronden die helpen om de benodigde verandering te verlichten. Daarin zou ik als belangrijkste richtlijn basiskennis over netwerken en computerarchitectuur willen benoemen. Zeker waar het gaat om werken met gedistribueerde systemen en Cloudomgevingen is het essentieel dat je als ontwikkelaar conceptueel begrijpt hoe zaken als performance, schaalbaarheid en veiligheid de compositie van jouw product of dienst beïnvloeden. Goede voorbeelden van theoretisch inzicht van een systeem zijn bijvoorbeeld het CAP-theorema en de memory hierarchy. Als je eenmaal inziet dat data over een netwerk lezen/schrijven duidelijk andere performance levert dan wanneer je alles in-memory kan doen (en vooral waarom) dan leer je redeneren hoe je dit voor optimalisatie van jouw product of dienst kan benutten of waar er juist negatieve gevolgen ontstaan.

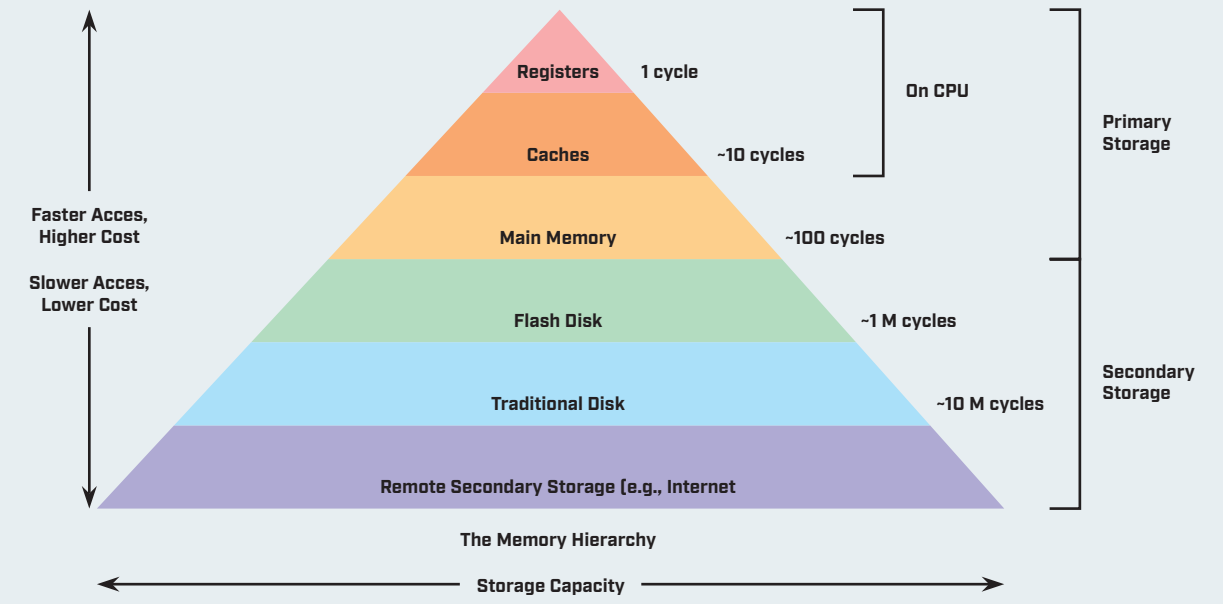
Betrouwbaarheid

Des te langer ik werk met gedistribueerde systemen en Cloud omgevingen, heb ik ervaren dat niet alles altijd werkt zoals verwacht of is gespecificeerd. De Cloud is immers "iemand anders zijn computer" en daar gaat ook wel eens wat mis. Binnen de context van één machine zijn we als Devs gewend dat alles eigenlijk altijd werkt. Er is nagenoeg geen latency en bestanden van disk lezen en schrijven lijkt altijd foutloos te gaan. Die illusie is lastig om los te laten maar door bijvoorbeeld in een Cloud omgeving met commodity hardware te gaan werken leer je dat niet elke call altijd succesvol is.
Dat leert je dat je zaken als reliability patterns, zoals bijvoorbeeld circuit breakers en retry policies, moet toepassen om de beschikbaarheid die je product of dienst levert te maximaliseren.

Kosten

Ook wordt duidelijk dat: "niets gratis is", en zeker niet in de Cloud. Het bewust worden van de kosten van dataverkeer over een netwerk en het aantal operaties dat je uitvoert tegen een object storage zijn

Wrijving tussen Dev en Ops is de uitdaging



Bron: Quora https://www.quora.com/What-is-Memory-hierarchy image: https://qph.cf2.quoracdn.net/main-qimg-7a90a61c09f30bc2ed08c4a806cf994a

nog diepere voorbeelden daarvan en bieden een geheel nieuwe uitdaging. Dat is een ander perspectief op dezelfde materie, dat wat ik het DevOps perspectief zou willen noemen. Het samenspel tussen softwareontwikkeling en de operationele aspecten voor het vinden van een optimale mix.
Als onderdeel van de continue optimalisatie kun je ook onderzoeken in hoeverre je de kosten optimaliseert. Dat kan door meer zelf te doen, of juist te vertrouwen op volledig beheerde omgevingen door de Cloud provider. Meer zelf doen bespaart in de maandelijkse kosten maar heeft als omgekeerd effect wel een groter beslag op tijd en affiniteit met het operationeel houden van de compute capaciteit vraagt. Het vergt daarnaast begrip over bijvoorbeeld verschillende typen storage en hoe deze te configureren zonder dat je de dienst downtime bezorgt als er een keer een disk vol loopt, of hoe je systemen zonder downtime kunt patchen.

Eigenaarschap

Ik zie de verandering van Dev naar DevOps zie als een verbreding van mijn vak, niet als compleet iets anders. Eigenlijk is het eigenaarschap nemen van die aspecten welke voorgaande generaties platform-ontwikkelaars en leveranciers jarenlang voor ons Devs hebben geprobeerd te beheren. Misschien is het een stap in het proces van volwassen worden aangezien nu wel duidelijk is dat dat verbergen van operationele complexiteit niet altijd de optimale uitkomst heeft opgeleverd. En zeker een operationele omgeving waar alle aspecten "software defined" zijn of APIs aanbieden, biedt bij uitstek voor Devs een omgeving waarin we prima uit de voeten kunnen. Dat helpt ook de organisatie waar we voor werken om sneller applicaties bij de klant geleverd te krijgen tegen vaak lagere kosten.

Continue verbeteren

De reis van Dev naar DevOps vervolgt zich eigenlijk langs de lijn van continue optimalisatie. Daarbij kunnen we voor individuele teams leunen op metrics die we als onderdeel van ons werk kunnen verzamelen. De DORA metrics (laten we daar maar een andere keer specifiek op terugkomen) bieden een wetenschappelijk verantwoorde blik op de kwaliteit van het proces waarop we software leveren aan gebruikers. Het is essentieel om deze goed te begrijpen want het biedt inzicht in de karakteristieken die een team kunnen blokkeren in hun groei en ontwikkeling. Ik denk wel dat deze metrics pas interessant zijn als een team enige tijd samenwerkt en de onderlinge rolverdeling heeft gevormd. Meer daarover is te lezen in het boek Acellerate van dr. Nicole Forsgren.
Het kunnen optimaliseren van processen leunt in veel gevallen op het gedachtegoed vanuit de wereld van Lean, wat je mogelijk kent vanuit manufacturing. Het ontwikkelen van softwareproducten en het ontwikkelen van fysieke producten aan een productielijn is niet altijd goed vergelijkbaar, en dus is niet alles uit dat gedachtengoed van toepassing op software. Echter de beginselen waarin value chain analysis en het analyseren van proces bottlenecks wordt beschouwd is weldegelijk waardevol omdat het statistisch inzicht geeft en aanleiding voor optimalisatie experimenten kan opleveren. Lees daarover meer in DevOps for the Modern Enterprise van Mirco Hering

Sustainability

Een prachtig voorbeeld van een nieuw aspect waarbij DevOps essentieel is, is het optimaliseren van de voetafdruk van een systeem. Hoeveel energie wordt er verbruikt en is dat uit verantwoorde bron, etc.? Er zijn daarbij nog heel veel vraagstukken onbeantwoord over welke metrics relevant en verantwoord bruikbaar zijn maar de algemene "groene aspecten" zijn belangrijk doel van vele organisaties die streven naar verantwoord ondernemen. Het kunnen bieden van inzicht daarin en het optimaliseren is dus goed voor de hele organisatie, en uniek in scope voor DevOps.

Niet per definitie agile

Een bestaande misvatting in mijn opinie is dat DevOps nogal eens verward wordt met Agile. Dat is eigenlijk appels met peren vergelijken. Echter, het uitgangspunt om continue high quality software te leveren waarbij je snel op veranderingen in kunt spelen heeft wel overeenkomsten met DevOps. Dit is bijvoorbeeld waar practices zoals continuous delivery/deployment en continuous improvement om de hoek komen kijken. De essentie daarvan is dat je als ontwikkelaar zo snel mogelijk feedback wilt hebben op features die je ontwikkelt. De manier waarop dat gestalte krijgt is door middel van het snel kunnen leveren van de software aan de gebruikers door middel van een pipeline.

Feedback

Het ontwikkelen van die pipeline, het daarin opnemen van de benodigde quality gates en build breakers is essentieel om de feedback loop naar de gebruikers te creëren. Daarnaast moet de software inzicht bieden in het gebruik, want hoe weet je immers

BIO

Carlo Kuip

Carlo Kuip werkt als senior consultant voor Microsoft met specialisatie "app innovation and modernization". Je kunt hem op twitter vinden: @CarloKuip en zijn persoonlijk blogs zijn te lezen op https://www.carlokuip.me

of de software die je uitrolt beter is als de voorgaande? Dat brengt ons wederom bij inzicht in operationele aspecten, performance, errors, latency, feature gebruik en gebruiksgemak, het zijn allemaal factoren die we automatisch kunnen verzamelen om inzicht te bieden en een nog beter product of dienst te bouwen.

Incrementeel bouwen

Dat vereist wel een specifieke manier van een systeem bouwen, namelijk dat een systeem incrementeel geleverd moet kunnen worden. Daarin moet je dus niet alleen nadenken over initiële deployability maar er ook rekening mee houden dat bijvoorbeeld uitbreidingen moeten kunnen worden toegevoegd en sommige dingen niet direct verwijdert kunnen worden, etc. Er zijn ondertussen allerlei goed gedocumenteerde patterns en practices beschikbaar voor het incrementeel bouwen en uitrollen van software bekend. Het is vervolgens een kwestie van onderzoeken welke aspecten en eigenschappen het best overeenkomen met het ontwikkelen van jouw dienst of product en die practices waar nodig te adopteren.

Infrastructure as code

Het incrementeel ontwikkelen van de infrastructuur waarop je applicatie of dienst draait is zeker met het gebruik van Cloud volledig te automatiseren. Er bestaan verschillende manieren om dat te doen en de voor mij meest gebruikelijke is met het gebruik van "Infrastructure as Code". Daarbij kun je of een declaratieve of een meer procedurele aanpak volgen. Uiteindelijk gebruik je een software defined datacenter om te definiëren hoe een mix van compute, storage en networking met elkaar verbonden zijn.
Ik vermoed dat de meeste Devs in de verandering naar DevOps het meeste opzien tegen het nemen van deze stap. Dit inderdaad het moment waarop je kennis en kunde over infrastructuur moet vertalen in specificaties of API-calls en moet leren te begrijpen welke gevolgen dat heeft voor de infrastructuur die je opbouwt en de applicatie die daar al bovenop draait. Dat vereist het ontwikkelen van affiniteit met die aspecten. De enige manier die ik ken om dat te leren is door er hands-on mee te werken, gewoon doen.
Misschien is het ook wel "angst voor het onbekende" want veel van de practices zijn helemaal niet nieuw. Traditionele operations medewerkers hadden al goede practices ontwikkeld en die kunnen we gewoon overnemen. Daarmee kunnen we er bijvoorbeeld voor zorgen dat we wijzigingen veilig kunnen testen voordat we deze op onze productieomgeving toepassen. Dat is ook het mooie van IaC, je kunt infrastructuur



met het grootste gemak definiëren, configureren en ook weer weggooien na gebruik. Door dit proces ook in een pipeline te vatten beperk je tevens de kans op fouten door menselijk handelen waardoor de kwaliteit van de wijziging nog verder toeneemt.

Blijven leren

De weg van Dev naar DevOps is best lang, dat zal ik zeker niet ontkennen maar dat betekent niet dat de weg altijd ongemakkelijk is. Als Dev word je in eerste instantie geconfronteerd met veel zaken die buiten je comfort zone zullen liggen maar gaandeweg krijg je daar grip op. De beste om vooruit te komen in die reis, is vol vertrouwen in de nieuwe materie te duiken en te weten dat je als ontwikkelaar het lerend vermogen hebt om dit allemaal stapje-voor-stapje te kunnen leren beheersen.

Goed gedocumenteerde patterns en practices voor het incrementeel bouwen en uitrollen van software beschikbaar

Het schalen van Azure App Service

Maak kennis met Azure App Service, een juweeltje in de Azure-servicecatalogus dat snel een van mijn favorieten is geworden. Dit krachtige platform stelt je in staat om naadloos je lift-and-shift applicaties over te zetten van een virtuele machine naar een zeer gepolijste, semi-cloud-native app. Onder de vele indrukwekkende functies biedt Azure App Service zowel verticale als horizontale automatische schalingsmogelijkheden. Dit betekent dat je indien nodig rekencapaciteit op aanvraag kunt benutten of kosten kunt besparen tijdens periodes van inactiviteit. Maar hoe zet je deze automatische schalingsfunctionaliteit eigenlijk op? Welke metrics, instellingen en strategieën zijn essentieel voor een succesvolle implementatie? Laten we duiken in de wereld van Azure App Service auto-scaling.

Auteur: Twan Koot

Basisprincipes van schalen

Schalen is een cruciaal aspect van het beheren van computing power in een cloudomgeving, en er zijn twee primaire opties om te overwegen: horizontaal schalen en verticaal schalen. Horizontaal schalen houdt in dat er compute units, zoals VM's of containers, van dezelfde grootte of type worden toegevoegd of verwijderd om wisselende workloads aan te kunnen. Een eenvoudig voorbeeld is het inzetten van nieuwe instanties van een app naast bestaande exemplaren of deze naar behoefte te verwijderen. Deze aanpak biedt een flexibelere en kosteneffectievere manier om schommelingen in de vraag op te vangen.

Aan de andere kant richt verticaal schalen zich op het vergroten van de rekenkracht van een enkele node of VM. Een veelvoorkomend voorbeeld is het vergroten van de vCPU-cores op een VM of pod om de verwerkingscapaciteit te vergroten. Deze methode kan met name nuttig zijn wanneer specifieke componenten binnen een applicatie meer resources nodig hebben om optimaal te presteren.
Zowel horizontale als verticale schaalopties voorzien in verschillende behoeften en scenario's, waardoor het essentieel is om hun unieke voordelen en beperkingen te begrijpen om je cloudomgeving effectief te optimaliseren.

Schalingsopties

Azure App Service biedt drie verschillende schaalconfiguraties om te gebruiken voor diverse scenario's. Elk van deze opties kan je helpen jouw omgeving te optimaliseren voor prestaties, kosten en betrouwbaarheid.

- Handmatig schalen: Deze optie is ideaal voor testdoeleinden of de initiële opzet van de omgeving. Het is echter niet aan te bevelen voor productieomgevingen, omdat het handmatige interventie vereist om resources aan te passen, wat tijdrovend kan zijn en vatbaar is voor menselijke fouten.

- Gepland schalen: Met deze methode is het mogelijk vaste data of tijden in te stellen voor het in- of uitschalen, waardoor het een handige en laagdrempelige oplossing is voor veel organisaties. Door te profiteren van voorspelbare gebruikspatronen, kan gepland schalen leiden tot zowel prestatieverbeteringen als kostenvermindering. Een goed voorbeeld is het terugschalen van de infrastructuur tijdens de nachtelijke uren of in het weekend wanneer het gebruik doorgaans lager is. Deze benadering zorgt ervoor dat resources efficiënt worden toegewezen en tegelijkertijd optimale prestaties worden gehandhaafd tijdens piekuren.

- Automatisch schalen: De laatste optie is automatisch schalen, dat het schalingsproces automatiseert op basis van metrics of events. Door automatisch schalen in uw omgeving op te nemen, kunt u het meeste uit de Azure omgeving halen door efficiënt hoge piekmomenten te bedienen en tegelijkertijd een kosteneffectieve opstelling te behouden. Deze dynamische methode past resources in realtime aan om

aan de eisen van uw applicatie te voldoen, en maximaliseert zowel prestaties als kostenefficiëntie.

Bij het configureren van Azure App Service is het essentieel om de juiste schaalinstelling te kiezen op basis van specifieke behoeften en gebruikspatronen.

USE RED

Als je automatisch schalen op basis van metrics en events wilt implementeren, is het belangrijk om te beginnen met de juiste basis. Ten eerste, concentreer op het verticaal schalen door de juiste instantiegrootte te selecteren. De reden hiervoor is dat het wijzigen van de size van een app-instance een herstart vereist, wat de beschikbaarheid kan beïnvloeden. Om de juiste size te bepalen kun je de volgende stappen volgen:
Load Testing: Begin met het testen van de applicatie onder verschillende belastingen om de prestaties en resourcebehoeften te begrijpen. Hierdoor krijg je waardevolle informatie over hoe de app zich gedraagt onder verschillende omstandigheden en kan je onderdelen identificeren die geoptimaliseerd moeten worden.
Optimalisatie: Analyseer de resultaten van de load testing en voer indien nodig optimalisaties uit om de prestaties en efficiëntie van de applicatie te verbeteren. Denk bijvoorbeeld aan het verfijnen van de code, het optimaliseren van databasequery's of het benutten van caching-technieken.
Selecteer een geschikte grootte: Kies op basis van de testen en optimalisatie-aanpassingen een geschikte instantiegrootte voor de app. Je kan bijvoorbeeld kiezen voor een P1V3-instantie. Onthoud dat het over het algemeen raadzaam is om te kiezen voor de kleinst mogelijke sizing, omdat dit meer controle biedt bij het schalen en grotere kostenbesparingen kan opleveren in vergelijking met het altijd vertrouwen op grotere VM's.

Door deze stappen te volgen, wordt er een solide basis gelegd voor het automatisch schalen op basis van metrics en events.
Om automatische schaling regels effectief te implementeren binnen Azure, is het belangrijk om een applicatie te monitoren met de juiste aanpak. Twee populaire monitoring-methodologieën kunnen helpen bij het opzetten van schaalregels voor het op- en afschalen, waarbij het laatste cruciaal is voor het verminderen van kosten. De eerste methodologie is USE, wat staat voor Utilization (benutting), Saturation (verzadiging) en Errors (fouten).
Utilization (U): het deel van de resource dat wordt gebruikt, dus 100% benutting betekent dat er geen werk meer kan worden geaccepteerd;
Saturation (S): de mate waarin een hardware resource extra werk heeft dat het niet kan verwerken, waardoor een wachtrij ontstaat voor deze resource;
Errors (E): het aantal foutevents;
Hoewel de USE-methodologie een waardevol startpunt biedt voor automatische schaalregels door middel van resource-monitoring, is het misschien niet de meest effectieve aanpak voor het optimaliseren van kosten en gebruikerservaring. Door te concentreren op gebruikersgerichte metrics, kunnen efficiëntere regels ingesteld worden die beter aansluiten bij de werkelijke behoeften van gebruikers.
Hoewel infrastructuurmetrics belangrijk zijn om problemen zoals "out-of-memory" fouten te detecteren , sluit een gebruikersgerichte benadering beter aan bij het uiteindelijke doel van het bieden van een optimale gebruikerservaring. Een gewaagde uitspraak om dit perspectief te benadrukken is: "Ik vind het niet erg om 100% CPU- of geheugengebruik te hebben, zolang gebruikers er niet door worden beïnvloed; ik gebruik alles waarvoor ik heb betaald!"
Door gebruikerservaringsmetrics te prioriteren, kan je betere beslissingen nemen over wanneer op- of af te schalen, zodat resources efficiënt worden toegewezen en tegelijkertijd een hoogwaardige ervaring voor de gebruikers wordt gegarandeerd.
Een goede manier om dit soort metrics toe te passen, is door de RED-methodologie te gebruiken die voortkomt uit Google's SRE-aanpak. RED is afgeleid van de Four Golden Signals, wat ook een goede manier is om elk op metrics gebaseerd systeem op te bouwen.
Snelheid (R): Het aantal verzoeken per seconde.
Fouten (E): Het aantal mislukte verzoeken.
Duur (D): De tijd die nodig is om een verzoek te verwerken.

BIO

Twan Koot

Mijn naam is Twan Koot, 30 jaar oud, en ik ben werkzaam als Principal Expert op het gebied van Microsoft Azure bij Ordina. Ik ben gespecialiseerd in het optimaliseren van applicatie performance in de cloud en het terugdringen van kosten. Naast mijn rol als Principal Expert, vervul ik ook de functie van Codesmith bij Ordina. Dit houdt in dat ik regelmatig optreed als internationaal spreker op diverse podia om mijn kennis en ervaringen te delen.



In mijn optiek is duur een zeer effectieve metric kan zijn voor schaalbeslissingen, voornamelijk omdat het nauw verbonden is met de gebruikerservaring. Uitsluitend vertrouwen op foutgebaseerde metrics, zoals HTTP 503, kan reactief zijn in plaats van proactief, omdat het te laat kan zijn om het probleem met schalen op te lossen tegen de tijd dat de fout optreedt. Op dezelfde manier kan het gebruik van snelheidsgebaseerde metrics uitgebreide prestatietests vereisen om het exacte punt te bepalen waarop de gebruikerservaring verslechtert.

Duur, aan de andere kant, is een intuïtievere en proactievere metric die u in staat stelt om een consistent performance voor elke gebruiker te handhaven. Door u te concentreren op responstijden, kunt u ervoor zorgen dat extra capaciteit precies wordt gebruikt wanneer dat nodig is. Bovendien is duur een metric die gemakkelijk te begrijpen is voor zowel technische als niet-technische teamleden, waardoor het eenvoudiger wordt om te communiceren en verwachtingen vast te stellen. Door vragen te stellen als "Hoe lang zou een gebruiker moeten wachten op een reactie?", kunt u duidelijke doelen stellen en beter geïnformeerde beslissingen nemen over wanneer op- of af te schalen, wat uiteindelijk leidt tot een betere gebruikerservaring.

Hoe meer zielen, hoe meer vreugd

Voor een robuuste en proactieve schaalimplementatie is het belangrijk om responstijd te combineren met andere relevante metrics en inzichten uit loadtesten. Deze benadering stelt ontwikkelteams in staat om prestatiepieken of -degradatie nauwkeuriger te voorspellen, waardoor effectievere schaalbeslissingen mogelijk zijn.

Aangezien het opstarten van nieuwe instanties tijd kost (van enkele seconden tot minuten), is het kunnen voorspellen van aankomende werklastpieken essentieel om optimale prestaties te behouden. Evenzo is het belangrijk om rekening te houden met het afschalen op basis van meerdere metrics om mogelijke prestatievermindering te voorkomen. Terwijl een korte periode van lage responstijd bijvoorbeeld kan suggereren dat bronnen kunnen worden verkleind, kan het totale aantal verzoeken per minuut aangeven dat prestatieproblemen waarschijnlijk zijn als het verkeer toeneemt. In dergelijke gevallen kan het verstandig zijn om extra instanties langer te behouden om mogelijke verkeerspieken op te vangen. Door een uitgebreide benadering te hanteren die rekening houdt met meerdere metrics en inzichten uit prestatietests, kunt u een effectievere schaalimplementatie creëren die optimale prestaties en gebruikerservaring garandeert en tegelijkertijd kosteneffectief blijft.

Azure App Service is een krachtig platform dat organisaties in staat stelt om hun applicaties naar de cloud te brengen, inclusief schaalbaarheid, kosteneffectiviteit en prestatie optimalisatie.

Conclusie

Azure App Service is uitgegroeid tot een krachtig platform dat organisaties in staat stelt om hun applicaties efficiënt naar de cloud over te brengen, terwijl het schaalbaarheid, kosteneffectiviteit en prestatie optimalisatie biedt. Om optimaal gebruik te maken van de automatische schalingsmogelijkheden, is het cruciaal om de verschillende schalingsopties en configuraties te begrijpen, evenals de metrics en strategieën.

Door te concentreren op zowel infrastructuur- als gebruikersgerichte metrics, kunnen ontwikkelteams robuustere en proactieve schaalimplementaties creëren die inspelen op de werkelijke behoeften van gebruikers en tegelijkertijd optimale prestaties behouden.

Het implementeren van automatische schaling met Azure App Service vereist een solide basis van horizontale schaling en de juiste instantiegrootte, evenals een goed doordachte schalingsregels die is afgestemd op de behoeften en gebruikerspatronen van de organisatie. Door zorgvuldig de balans te vinden tussen prestaties, kosten en betrouwbaarheid in de cloudomgeving, kunnen organisaties ervoor zorgen dat ze de voordelen van Azure App Service kunnen maximaliseren. ●

DEVITJOBS.NL

Full Stack Engineer

Salarisindicatie:
60.000 - 90.000 EUR

Locatie:
Amsterdam

Technologieën:
Fullstack, JavaScript, C#, Java, Python, PHP, Ruby, Angular, Vue, React, TypeScript, NodeJS Svelte, .NET Framework, Git



.NET Developer

Salarisindicatie:
60.000 - 85.000 EUR

Locatie:
Nieuwegein

Technologieën:
.NET MVC, ASP.NET, .NET Core, .NET Framework, Azure, Backend, C#, Cloud, DevOps, MS-SQL, SQL, Windows Forms, Web, .NET

ORDINA

BI test expert 'Reisbegeleiding & Calamiteiten' – Ut

Salarisindicatie:
40.000 - 70.000 EUR

Locatie:
Utrecht

Technologieën:
Azure, CI/CD, Cloud, Data Warehouse, DevOps, H2, Informatica, ISTQB, Power BI, Product Owner, SQL, Snowflake, Spring, TMap, IT

Senior Software Developer

Salarisindicatie:
45.000 - 75.000 EUR

Locatie:
Delft

Technologieën:
Java, C++, C#, Cloud, DevOps, Docker, Vue, React, Angular, JavaScript

TOPdesk
Your guides to service excellence

Find all these vacancies and more

at Devitjobs.nl

ORDINA
Software Development

Ordina is trots om de locatiesponsor van Azure Fest te zijn

Kom op 13 september naar het hoofdkantoor van Ordina in Nieuwegein om je kennis met betrekking tot Azure uit te breiden

Azure Fest NL is een gratis eendaags community event omtrent Azure

Azure Fest brengt lokale en internationale sprekers van wereld-klasse samen die praktijkervaringen, best practices en de laatste ontwikkelingen op het Azure platform delen.



Gratis tickets vanaf 1 juli verkrijgbaar op **www.azurefest.nl**