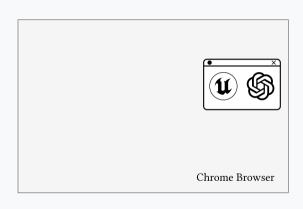
DarthShader: Fuzzing WebGPU Shader Translators & Compilers

Lukas Bernhard, Nico Schiller, Moritz Schloegel, Nils Bars, Thorsten Holz



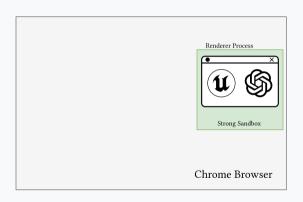
WebGPU - Exposing GPUs to the Web



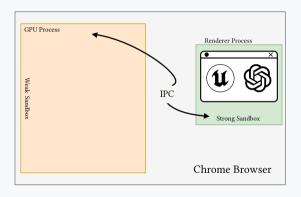


WebGPU - Exposing GPUs to the Web

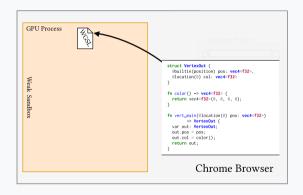




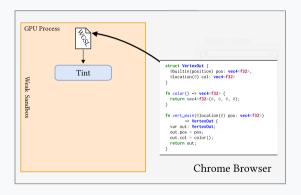




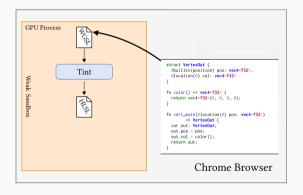




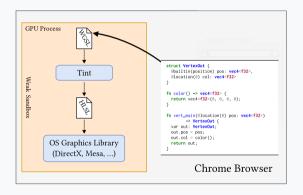




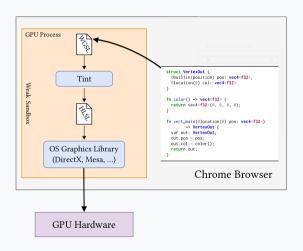




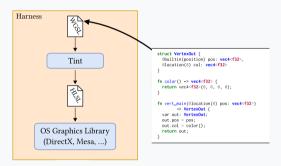






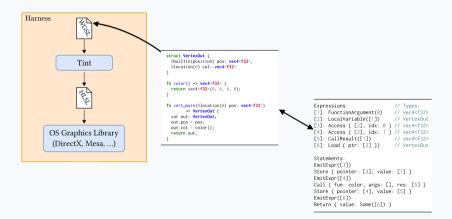






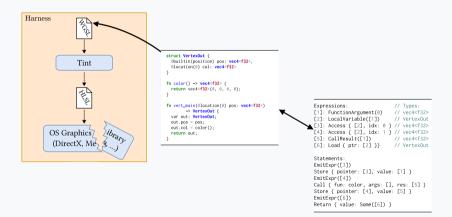
Fuzzing - IR Mutations



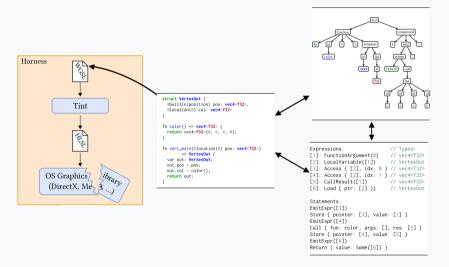


Fuzzing - IR Mutations

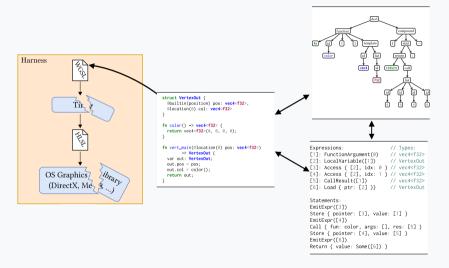




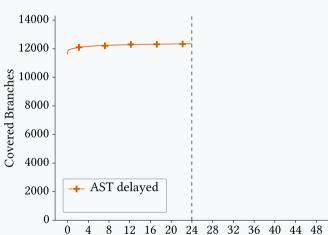






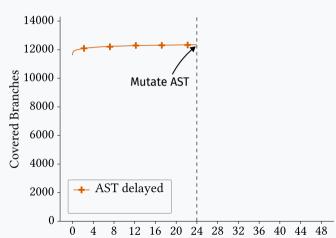




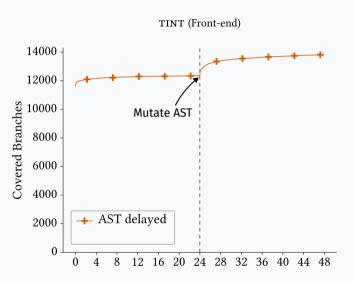


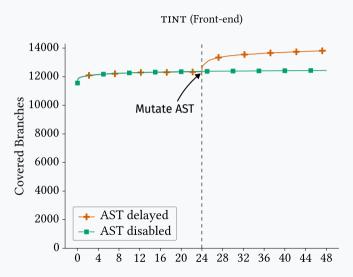














SUT	Bug ID	Browser	Status	
angle	chromium 329271490	ඉ ප @	fixed	
dxcompiler	chromium 1513069	9	open	
dxcompiler	CVE-2024-2885	Ø	fixed	
dxcompiler	CVE-2024-3515	Ø	fixed	
dxcompiler	CVE-2024-4948	9	fixed	
dxcompiler	CVE-2024-4060	9	fixed	
dxcompiler	CVE-2024-4368	Ø	fixed	
dxcompiler	CVE-2024-5160	Ø	fixed	
dxcompiler	CVE-2024-5494	0 0 0	fixed	
tint	tint 2190	Ø	fixed	
tint	tint 2201	9	fixed	
tint	tint 2202	0	fixed	
tint	tint 2055	Ø	fixed	
tint	tint 2056	Ø	fixed	
tint	tint 2058	9	fixed	
tint	tint 2068	Ø	fixed	
tint	tint 2076	Ø	fixed	
tint	tint 2077	9	fixed	
tint	tint 2078	9	fixed	
tint	tint 2079	Ø	fixed	

Bugs, Bugs Everywhere



SUT	Bug ID	Browser	Status	SUT	Bug ID	Browser	Status
angle	chromium 329271490	9 U Ø	fixed	dxcompiler	CVE-2024-5495	9	fixed
dxcompiler	chromium 1513069	Ø	open	dxcompiler	CVE-2024-6102	Ø	fixed
dxcompiler	CVE-2024-2885	Ø	fixed	dxcompiler	CVE-2024-5831	Ø	fixed
dxcompiler	CVE-2024-3515	9	fixed	dxcompiler	CVE-2024-5832	9	fixed
dxcompiler	CVE-2024-4948	9	fixed	dxcompiler	CVE-2024-6290	9	fixed
dxcompiler	CVE-2024-4060	9	fixed	dxcompiler	CVE-2024-6292	Ø	fixed
dxcompiler	CVE-2024-4368	9	fixed	dxcompiler	CVE-2024-6103	9	fixed
dxcompiler	CVE-2024-5160	9	fixed	dxcompiler	CVE-2024-6293	9	fixed
dxcompiler	CVE-2024-5494	9	fixed	dxcompiler	CVE-2024-6991	9	fixed
tint	tint 2190	9	fixed	tint	tint 2092	Ø	open
tint	tint 2201	Ø	fixed	tint	tint 2194	Ø	open
tint	tint 2202	9	fixed	naga	naga 2560	೮	fixed
tint	tint 2055	9	fixed	naga	naga 2568	೮	fixed
tint	tint 2056	Ø	fixed	naga	wgpu 4547	೮	open
tint	tint 2058	9	fixed	naga	wgpu 4512	೮	open
tint	tint 2068	Ø	fixed	naga	wgpu 4513	೮	open
tint	tint 2076	9	fixed	naga	wgpu 5547	೮	fixed
tint	tint 2077	Ø	fixed	wgslc	webkit 268148	Ø	open
tint	tint 2078	9	fixed	wgslc	webkit 273407	Ø	fixed
tint	tint 2079	Ø	fixed	wgslc	webkit 273411	Ø	fixed



GPU stack is exposed by WebGPU

DarthShader: Fuzzing WebGPU Shader Translators & Compilers

Lukas Barnhard Nico Schiller Mority Schlosgal CISPA Helmholtz Center for CISPA Helmholtz Center for CISPA Helmholtz Center for Information Security Information Sacurity Information Security Germany Germany lukas hemhard@cispa.de nico.schillen@cispa.de moritz.schloegel@cispa.de Nils Bars Thorsten Holz

CISPA Malmholty Center for Information Security nils bars@cispa.de

ABSTRACT

A recent trend towards running more demanding web applications, such as video games or client-side LLMs, in the browser has led to the adoption of the WebGPU standard that provides a crossplatform API exposing the GPU to websites. This opens up a new ottack surface. Untrusted such content is ressed through to the CPS stack, which traditionally has been optimized for performance instead of security. Worsening the problem, most of WebGPU cannot be run in the tightly sandboxed process that manages other web content, which eases the attacker's path to compromising the client machine. Contrasting its importance. WebGPU shader processing has received summissionly little attention from the automated test ing community. Part of the reason is that shader translators consect highly structured and statically typed input, which renders typical fazzing mutations ineffective. Complicating testing further, shader translation consists of a complex multi-sten compilation pineline. each stage presenting unique requirements and challenges. In this paper, we propose DARTHSHADER, the first language

fuzzer that combines mutators based on an intermediate representotion with those using a more traditional abstract evetor tree. The key idea is that the individual stages of the shader compilation nineline are supportible to different classes of faults, requiring entimbs different restation strategies for thereweb testing the furnion the full pipeline, we ensure that we maintain a realistic attacker model. In an empirical evaluation, we show that our method outperforms the state-of-the-art fuzzers regarding code coverage. Furthermore, an extensive ablation study validates our key design. DARTUSHADER found a total of 39 software faults in all modern Introduces... Charges. Finefey, and Safari... that price week missed. For 15 of them, the Chrome team assigned a CVE, acknowledging the irreact of our rambs.

This is the author's version of the week. It is nosted here for your personal use. Not for redistribution. The definitive Version of Record year published in ACM Conference on Computer and Communications Security (CCSL https://dx.doi.org/10.1145/3658644.3690309.

CCS CONCEPTS

· Security and privacy -- Browser security: Systems security: • Computing methodologies -- Grankics systems and interfaces.

holmiteisna de KEYWORDS

Germany

CISPA Helmholty Contex for

Information Security

Fuzzinat. Software Security. Beowner Security. Graphics Shaders. WebGPU, WGSL

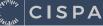
Lakas Berahard, Nico Schiller, Maritz Schloovel, Nils Burs, and Thorsten. Hels. 2024. DurthShador: Fuzzing WebGPU Shader Translators & Compilers. In Proceedings of ACM Conference on Computer and Communications Security

1 INTRODUCTION

The internet and the web have been game changers in the past decades enabling instant access to stobol news, constant connection with friends and accusintances and many tones of new hosiness models. Web browsers, in particular, play a crucial role in this ecosystem, as they are the most important applications to access the web for many users. However, the ubiquitous connectivity of the internet also enables adversaries with malicious intent, exposing sours to notential threats as they mayingte the web. A common security risk is memory sofety violations [50], which have been the starting point for many successful attacks in the nast

As a result, we require fundamental, proactive measures to improve defenses against such threats and strengthen web beowsers. against various attack vectors. By using handware-supported security features such as memory randomization (ASLR) and nonexecutable memory regions, web beowees can reduce the risk of exploits that attempt to execute arbitrary code. Moreover, ristorous testing poods to be performed on all browner components. This inchafes web APIs [16, 26] and boorferint entires [21, 23, 30, 43, 54]. given that they are often terreted due to their complexity and the fine-grained control they expose to adversaries. In addition, sandlyxive is a crucial defense mechanism designed to prevent code from performing mulicious actions or accessing sensitive data outside its intended scope [14, 37]. This technique enforces a strict separation between the content of different websites in different processes (called airc isolation [41]) and most importantly between web content and the privileged components of the because a nthose with access to the file review. Technically encolving sandbox ing is implemented by executing code of different sites in separate processes with restricted authorizations. Each process is confined by a security policy enforced at the operating system level, which

Takeawavs



- GPU stack is exposed by WebGPU
- GPU stacks are a weak spot
- and lack in-depth security testing

DarthShader: Fuzzing WebGPU Shader Translators & Compilers

Lukas Bernhard CISPA Helmholtz Center for Information Security Germany Inkas bernhard@cispa.de	Nico Schiller CISPA Helmholtz Center for Information Security Germany nico.schiller@cispa.de	Moritz Schloegel CISPA Helmholtz Center for Information Security Germany moritz.schloegel@cispa.de	
N	file Bare Thorst	ian Holy	

CISPA Malmholty Center for Information Security nils bars@cispa.de

ABSTRACT

A recent trend towards running more demanding web applications, such as video games or client-side LLMs, in the browser has led to the adoption of the WebGPU standard that provides a crossplatform API exposing the GPII to websites. This opens up a new ottack surface: Untroited such content is respect through to the CPU stack, which traditionally has been optimized for performance instead of security. Worsening the problem, most of WohGPU cannot be run in the tightly sandboxed process that manages other web content, which eases the attacker's path to compromising the client machine. Contrasting its importance. WebGPU shader processing has received summissionly little attention from the automated test ing community. Part of the reason is that shader translators expect highly structured and statically typed input which renders typical fuzzing mutations ineffective. Complicating testing further, shader translation consists of a complex multi-step compilation vineline. each stage presenting unique requirements and challenges. In this paper, we propose DARTHSHADER, the first language

fuzzer that combines mutators based on an intermediate representotion with those using a more traditional abstract eventure tree. The key idea is that the individual stages of the shader compilation nineline are supportible to different classes of faults, requiring entimbs different restation strategies for thereweb testing the furnion the full simpling are ensure that we maintain a realistic offerhor model. In an empirical evaluation, we show that our method outperforms the state-of-the-art fuzzers regarding code coverage. Furthermore, an extensive ablation study validates our key design. DARTUSHADER found a total of 39 software faults in all modern https://granus.Christine.Firefox. and Safari...that prior work missed. For 15 of them, the Chrome team assigned a CVE, arknowledging the irroact of our rambs

This is the author's version of the week. It is nosted here for your personal use. Not for redutribution. The definitive Version of Record year published in ACM Conference on Computer and Communications

CCS CONCEPTS

· Security and privacy -- Browser security: Systems security: • Computing methodologies -- Grankics systems and interfaces.

Information Security holmiteisna de KEYWORDS

CISPA Helmholtz Center for

Fuzzine. Software Security. Beowart Security. Graphics Shaders WebGPU, WGSL

Lakas Berahard, Nico Schiller, Maritz Schloovel, Nils Burs, and Thorsten. Hels. 2024. DurthShador: Fuzzing WebGPU Shader Translators & Compilers. In Proceedings of ACM Conference on Computer and Communications Security

1 INTRODUCTION

The internet and the web have been gome changers in the past decades enabling instant access to stobol news, constant connection with friends and accusintances, and many tones of new lossiness models. Web browsers, in particular, play a crucial role in this ecosystem, as they are the most important applications to access the web for many users. However, the ubiquitous connectivity of the internet also enables adversaries with malicious intent, exposing sours to notential threats as they mayingte the web. A common security risk is memory sofety violations [50], which have been the starting point for many successful attacks in the nast

As a result, we require fundamental, proactive measures to im prove defenses against such threats and strengthen web beowsers. against various attack vectors. By using handware-supported security features such as memory randomization (ASLR) and nonexploits that attempt to execute arbitrary code. Moreover, ristorous testing poods to be performed on all browner components. This inchales with APR [16, 26] and SwaScript engines [21, 21, 30, 43, 54] given that they are often tograted due to their complexity and the fine-grained control they expose to adversaries. In addition, sandfourty is a crucial defense mechanism designed to present code from performing mulicious actions or accessing sensitive data outside its intended scope [14, 37]. This technique enforces a strict separation between the content of different websites in different processes (called airc isolation [41]) and most importantly between with content and the privileged components of the because a a those with access to the file review. Technically encolving sandbox ing is implemented by executing code of different sites in sense atprocesses with restricted authorizations. Each process is confined by a security policy enforced at the operating system level, which

Takeaways



- GPU stack is exposed by WebGPU
- GPU stacks are a weak spot
- and lack in-depth security testing
 - Combination of IR and AST fuzzing
- exposes multitude of security issues

DarthShader: Fuzzing WebGPU Shader Translators & Compilers

Lukas Bernhard	Nico Schiller		Moritz Schloegel
A Helmholtz Center for	CISPA Helmholtz Center for		CISPA Helmholtz Center for
aformation Security	Information Security		Information Security
Germany	Germany		Germany
as bernhard@cispa.de	nico.schiller@cispa.de		moritz.schloegel@cispa.de
Nils Bars CISPA Helmholtz Center for		Thorste CISPA Helmh	

Information Security Germany nils.bars@cispa.de

ABSTRACT

A recent frend towards running more demanding web applications, such as video games or client-side LLMs, in the browser has led to the adoption of the WebGPU standard that provides a crossplatform API exposing the GPU to websites. This opens up a new ottack surface: Untroited such content is respect through to the CPU stack, which traditionally has been optimized for performance instead of security. Worsening the problem, most of WohGPU cannot be run in the tightly sandboxed process that manages other web content, which eases the attacker's path to compromising the client machine. Contracting its importance. WebGPU shader processing has received summissionly little attention from the automated test ing community. Part of the reason is that shader translators expect highly structured and statically typed input which renders typical fuzzing mutations ineffective. Complicating testing further, shader translation consists of a complex multi-step compilation pipeline. each stage presenting unique requirements and challenges. In this paper, we propose DARTHSHADER, the first language

future that confinent mainten based on an intermediate experient more with these users must testimate abstract versity to the Tation with the confinence of the confinence of the confinence of the pipeline are susceptible to different cutsors of future version as a confinence of the confinence of the confinence of the confinence of the theory of the confinence of the confinence of the confinence of the displacement of the confinence of the

This is the author's version of the work. It is posted here for your personal use. Not for reductibation. The definitive Version of Record-way published in ACM Conference on Computer and Continuations Security (CCS), https://dx.doi.org/10.1245/3606044.359009.

CCS CONCEPTS

Security and privacy → Browser security; Systems security;
 Computing methodologies → Graphics systems and interfaces.

Information Security
Germany
holo@cispa.de

KEYWORDS

Fuzeing, Software Security, Beowser Security, Graphics Shaders, WebCPU, WGSL

ACM Reference Forms

Lakas Berahard, Nico Schiller, Meritz Schloegel, Nils Bars, and Thoesten Hols, 2004. DarthMader Fuzzing WelGPU Shader Translation & Compilers in Proceedings of ACM Conference on Computer Act Consumerication Security (CCS '24). ACM, New York, NY, USA, 13 pages, https://doi.org/19.1145/

1 INTRODUCTION

The interest and the web have been game changers in the past decises, enabling instanceases juddenians, constant connection with friends and acquaintances, and sumy types of new business models. Web however, in particular, judy a created rele is the ecception, as they are the most important applications to access the web for army unreal. Instern, the eliquion connectivity of the interest also madels adversaries with subinious interes, reposing were to puterful allowards as they acceptable the web. A current or control of the control of the control of the control of the acceptable problem or my proceeding lands to his part of the part of a best particular and proceeding lands to his part of the part of the particular and the p

As a result, we require fundamental, proactive measures to im prove defenses against such threats and strengthen web browsers against various attack vectors. By using handware-supported security features such as memory randomization (ASLR) and nonexploits that attempt to execute arbitrary code. Moreover, ristorous testing poods to be performed on all browner components. This inchales with APR [16, 26] and SwaScript engines [21, 21, 30, 43, 54] given that they are often terreted due to their complexity and the fine-grained control they expose to adversaries. In addition, sand Society is a crucial defense mechanism designed to prevent code from performing mulicious actions or accessing sensitive data out side its intended scope [14, 37]. This technique enforces a strict separation between the content of different websites in different processes (called airc isolation [41]) and most importantly between with content and the privileged components of the because a a those with access to the file review. Technically encolving sandbox ing is implemented by executing code of different sites in separate processes with restricted authorizations. Each process is confined by a security policy enforced at the operating system level, which



- GPU stack is exposed by WebGPU
- GPU stacks are a weak spot
- and lack in-depth security testing
- Combination of IR and AST fuzzing
- exposes multitude of security issues



github.com/wgslfuzz/darthshader

DarthShader: Fuzzing WebGPU Shader Translators & Compilers

Lukas Bernhard SPA Helmholtz Center for Information Security Germany	Nico Sci CISPA Helmhol Information Germa	z Center for Security ny	Moritz Schloegel CISPA Helmholtz Center for Information Security Germany
ikas.hemhard@cispa.de	nico.schiller(moritz.schloegel@cispa.de
Nils Bars CISPA Helmholty Center for		Thorsto CISPA Helmh	

Information Security Germany nils.bars@cispa.de

ABSTRACT

A recent frend towards running more demanding web applications, such as video games or client-side LLMs, in the browser has led to the adoption of the WebGPU standard that provides a crossplatform API exposing the GPU to websites. This opens up a new ottack surface: Untroited such content is respect through to the CPU stack, which traditionally has been optimized for performance instead of security. Worsening the problem, most of WohGPU cannot be run in the tightly sandboxed process that manages other web content, which eases the attacker's path to compromising the client machine. Contracting its importance. WebGPU shader processing has received summissionly little attention from the automated test ing community. Part of the reason is that shader translators expect highly structured and statically typed input which renders typical fuzzing mutations ineffective. Complicating testing further, shader translation consists of a complex multi-step compilation pipeline. each stage presenting unique requirements and challenges. In this paper, we propose DARTHSHADER, the first language

trace that condom mattern based on an intermediate representation with these using some relationsh abstract system. See The lay likes a that the individual stages of the shader comprisions to the stage of the stage of the shader comprisions the relative stage of the stage of the shader comprisions the relative stage of the stage of the stage of the shader comprisions the full pripature, we ensure that we maintain a residue statute shaded in one required evidentistic, we show that cape the shader of shaded in the required evidentistic relative stage of the stage of the

This is the author's version of the work. It is posted here for your personal use. Not for reducerholdon. The definitive Ventica of Record was published in ACM Conference on Computer and Constructations Security (CCS), https://doi.oio.org/10.1145/3606043.590009.

CCS CONCEPTS

Security and privacy → Browser security: Systems security:
 Computing methodologies → Graphics systems and interfaces.

Information Security Germany holz@cispa.de

EYWORDS

Fuzeing, Software Security, Browser Security, Graphics Shaders WebCPU, WCSL

ACM Reference Form

Lukas Bernhard, Nico Schiller, Muritz Seltbergel, Nils Bars, and Thoesten. Hult. 2004. DarthMader Fuzzing WelGPU Shader Translation & Compilers. In Proceedings of ACM Conference on Computer and Communication Security. 8COS '249, ACM, New York, NY, USA, 13 pages, https://doi.org/10.1109/

1 INTRODUCTION

the interest and the web have been game changers in the part decades, earling instancement platforms, content convention while frainful and acquisitations, and many types of now luminous models. Web however, in particula, byte a restal rela is the ecosystem, as they are the most important applications to access the whole for many content and the acquisition concernity of the interent also establish subversate the shapinous concernity of the interent also establish subversate with multivision interest, reposing, ware to potential ulmara as they average the who. A common accusity risk is mercury sofely violatine [19], which have been the softing point for many secretally intacts in the past.

As a result, we require fundamental, proactive measures to im prove defenses against such threats and strengthen web browsers against various attack vectors. By using handware-supported security features such as memory randomization (ASLR) and nonexploits that attempt to execute arbitrary code. Moreover, ristorous testing poods to be performed on all browner components. This inchales with APR [16, 26] and SwaScript engines [21, 21, 30, 43, 54] given that they are often terreted due to their complexity and the fine-grained control they expose to adversaries. In addition, sand Society is a crucial defense mechanism designed to prevent code from performing mulicious actions or accessing sensitive data out side its intended scope [14, 37]. This technique enforces a strict separation between the content of different websites in different processes (called airc isolation [41]) and most importantly between with content and the privileged components of the because a a those with access to the file system. Technically menicing, sandbox ing is implemented by executing code of different sites in senseate processes with restricted authorizations. Each process is confined by a security policy enforced at the operating system level, which