# Research Statement

Kaiming Huang

Security-critical systems increasingly operate beyond settings in which end-to-end formal verification of whole-system correctness is practical. Comprehensive protection may be too expensive to apply uniformly, particularly for memory safety where existing defenses face persistent tradeoffs among coverage, compatibility, and runtime cost [1, 2]; the semantics needed for precise reasoning may be transformed or lost across language, compiler, and binary boundaries [3–5]; learned models can exhibit fragile behavior under carefully constructed perturbations [6, 7]; and testing can leave substantial residual uncertainty even after extensive exploration [8]. Rather than attempting to prove entire systems correct, my research designs security mechanisms with explicit guarantees and validates the conditions under which those guarantees hold. I develop validations that determine **what can be guaranteed, under what conditions, and what uncertainty remains** when end-to-end verification is infeasible. Rather than treating assurance as an all-or-nothing property, **my work combines program analysis, verification, and practical enforcement to establish deterministic guarantees where sound reasoning is possible, probabilistic guarantees where uncertainty is inherent, and enforceable conditions that preserve these guarantees in deployment**.

I established this validation-driven approach through my Ph.D. research on **practical comprehensive memory safety** [9], using validation to determine which program objects and operations already satisfy the required safety conditions and concentrating stronger protection on the remaining security-sensitive state [10, 11]. Building on this foundation, I extended the same principle beyond individual memory objects to system interfaces, sanitization, eBPF, unsafe Rust, speculative execution, and compartmentalization, where security depends on assumptions distributed across multiple software layers [12–18]. During my postdoctoral research, I further broadened this methodology beyond conventional software in two directions: to AI-enabled systems, where learned behavior and deployment uncertainty must be bounded and controlled; and to security analyses where evidence or program semantics are inherently incomplete. This work includes synthesizing **runtime-enforceable security envelopes** for learned models and translating model-level guarantees into actionable operating conditions for deployment [19–22]. More broadly, my collaborative work has examined the personalization, human factors, cognitive attacks, and deployment conditions that shape **trustworthy AI-enabled mixed-reality systems** [23–30]. I also extend the same validation-driven methodology to settings where evidence or semantics are incomplete, including **quantifying residual fuzzing risk** to guide subsequent testing decisions [31] and **recovering security-relevant semantics from binary software**. Across these efforts, validation serves as the bridge between rigorous reasoning and practical security: recover the semantics needed for a security decision, determine the conditions under which a guarantee holds, and make the remaining uncertainty explicit when a complete guarantee cannot be established.

## Ph.D. Research: Validation-Driven Practical Security for Software and Systems

My Ph.D. research established the foundation of my validation-driven approach to practical security through comprehensive memory safety [9]. Existing defenses address spatial, temporal, and type-related memory errors, but comprehensive protection remains difficult to deploy at practical cost, forcing tradeoffs among coverage, compatibility, and runtime overhead [1, 2]. **My work reframes this problem by determining which program objects and operations already satisfy the required safety constraints** and focusing stronger analysis and enforcement on the unresolved state that can still threaten the guarantee. This turns comprehensive memory safety from uniform protection into a problem of precisely identifying where stronger protection is actually necessary.

I first realized this approach in **DataGuard** [10], which combines static analysis with targeted symbolic execution to identify stack objects for which spatial, type, and temporal memory safety can be established comprehensively. DataGuard shows that more than 91% of evaluated stack objects are inherently safe, allowing most stack state to avoid costly protection and substantially reducing the cost of comprehensive memory safety. I subsequently demonstrated that the methodology scales to more than 1,200 Ubuntu packages and millions of stack objects in real software [32]. With this scalability established, I extended the same validation-driven principle to the substantially more challenging setting of heap memory in **Uriah** [11], where aliasing, dynamic allocation, and object lifetimes

make temporal safety much harder to establish statically. Uriah separates what can be validated from what must be enforced: it statically establishes spatial and type safety and uses a restricted allocation strategy to preserve temporal allocated-type safety. Together, DataGuard and Uriah enable comprehensive protection for the large majority of validated memory objects and states at low single-digit runtime overhead.

More importantly, these systems established a **reusable foundation for subsequent memory-safety research**: later defenses can build on validated safety properties rather than reasoning about every object and operation from scratch, allowing new analysis and enforcement to focus on the states, execution modes, and interfaces that introduce unresolved risk. This foundation shaped the broader trajectory of my research: a strong security guarantee need not protect every part of a program in the same way; validation can instead expose the **boundary between established safety and unresolved risk** and concentrate stronger enforcement where the guarantee actually remains vulnerable.

I next extended this principle from individual objects to guarantees that span multiple software components. My work on **eBPF memory safety** [12] showed that its memory-safety guarantee depends on assumptions shared across the verifier, helper functions, compiler, and Linux kernel, and identified cases in which those assumptions are not consistently preserved. At the same time, most eBPF memory operations could still be validated, with as few as 1.62% remaining unresolved. This suggests a practical path toward eBPF memory safety: retain static validation for the common case and concentrate stronger runtime validation or isolation where cross-layer assumptions fail.

The same principle has guided a sustained and productive line of collaborative research on memory sanitization, unsafe Rust, speculative execution, and software compartmentalization. My main contribution has been to identify the objects, pointers, operations, and state that actually require stronger protection: prioritizing security-critical C/C++ objects under limited protection budgets [13]; designing compact, flexible metadata schemes for comprehensive memory safety by classifying pointers according to the metadata granularity they require and preserving metadata consistency across calls to uninstrumented libraries through interface-level summarization; exploiting Rust ownership and lifetime semantics for selective sanitizer instrumentation [14]; distinguishing memory operations requiring sequential versus speculative defenses [15]; and identifying shared state and interface values that require validation across isolation boundaries [16–18], including a joint effort with MIT Lincoln Laboratory that integrates memory safety validation with HAKC [33] to eliminate the corresponding runtime compartment-validation checks [34, 35]. Although these systems address different threats, they share the same objective: identify the state, operations, and assumptions on which a security guarantee depends, establish safety where rigorous validation is possible, and focus stronger protection where it is not. Together, these efforts moved my research from identifying safe and unsafe memory objects to validating **security-critical operations, interfaces, and cross-layer assumptions**, providing the technical foundation for my broader validation-driven research agenda.

## Deterministic and Probabilistic Validation for AI-Enabled Systems

During my postdoctoral research, I extended my validation-driven methodology from conventional software to AI-enabled systems, where guarantees depend on both learned behavior and uncertain deployment conditions. Rather than asking only whether a property holds over a predefined input region, my work asks **what operating conditions can be trusted and at what assurance level**, and how those conditions can be enforced during deployment. This shifts verification from certifying a fixed input region for robustness to synthesizing runtime-monitorable conditions under which a required property holds, directly connecting model-level guarantees to deployment decisions.

I first developed this deployment-oriented view as part of the **DARPA VeriPro program**, where I developed the verification component for a Random Forest model that predicts cybersickness in mission-oriented mixed-reality training. I encoded the model as a mixed-integer linear program to verify its behavior over participants and exposure variables, then translated the resulting guarantees into a configurable screening map that exposes admissible operating conditions, assurance levels, and residual risk [21]. This allows mission planners to directly identify which participant and deployment conditions satisfy a required safety level. The resulting guarantee format was reviewed and accepted through the DARPA program's validation process, reinforcing a principle that now guides my work on intelligent systems: verification should produce **actionable operating conditions**, not only abstract certificates.

I generalize this principle further in **PolySafe** [22], which synthesizes runtime-enforceable security envelopes for recurrent neural networks. Conventional neural-network verification typically asks whether a property holds over a predefined region [36, 37], such as local robustness; PolySafe instead asks **what operating conditions should be enforced so that the property is guaranteed to hold**. It synthesizes such regions over runtime-monitorable or controllable variables and independently verifies each candidate. To handle nonlinear gate interactions and temporal dependencies in recurrent models [38–40], I developed multi-convex abstractions, property-guided refinement, and sensitivity-guided temporal reasoning, building on DeepPoly and Prover [39, 41]. Across evaluated benchmarks, PolySafe certifies deterministic regions covering 18.9–35.9% of the admissible input space, compared with only 0.10–0.24% for existing verifiers. These larger certified regions give systems substantially more flexibility in deployment while retaining a formal safety guarantee. When deterministic regions are too restrictive, PolySafe adds a probabilistic shell that expands the usable operating space while making the corresponding assurance level explicit.

Together, these efforts establish a deployment-oriented approach to validation for AI-enabled systems: identify enforceable conditions under which learned behavior can be trusted, provide deterministic guarantees where rigorous reasoning is feasible, and quantify assurance where uncertainty cannot be eliminated. This work provides the technical foundation for the broader AI-verification agenda I plan to pursue as a faculty member.

## Trustworthy Security Analysis under Incomplete Evidence

In parallel, my postdoctoral research has examined how security analysis can remain trustworthy when execution evidence or program semantics are incomplete. These settings create two complementary problems: incomplete observations leave uncertainty about what behavior may still be undiscovered, while incomplete semantics leave uncertainty about how observed program state should be interpreted. My approach is therefore to **quantify residual uncertainty** in the former case and recover only the security-relevant semantics needed in the latter, while explicitly distinguishing established facts from hypotheses that still require validation.

Fuzz testing illustrates the first challenge: executing millions of tests without finding new behavior does not reveal how much reachable behavior remains unexplored. Building on prior work on residual risk [8], my recent collaborative work develops a Bayesian framework that estimates both the likelihood of additional discoveries and the effort required to reach them. Across real-world FuzzBench targets, it reduces weighted prediction error by approximately 24%, while reducing underestimation to 0.04% [31]. This turns the absence of new discoveries from an ambiguous outcome into **quantitative evidence about the risk that may still remain**.

A complementary challenge arises when necessary semantics have been lost before analysis begins [3–5]. In binary software, compiler transformations and stripping can obscure how program state evolves and which inputs can trigger security-sensitive behavior. My current work recovers these relationships directly from binaries by identifying which memory locations encode persistent state, which inputs and operations modify that state, and which state conditions make privileged or security-sensitive code reachable. Rather than reconstructing a complete high-level representation, the analysis focuses on the **minimum semantics needed for a specific security question**, allowing subsequent testing and reasoning to target meaningful input–state combinations. It combines runtime evidence, static data-flow analysis, and selective symbolic reasoning while tracking each recovered fact as *observed*, *established*, or *inferred*, making clear which conclusions are supported and which still require validation.

## Future Research Agenda: Security Guarantees for Evolving and Uncertain Systems and Software

Building on these foundations, I will develop validation-driven methods that maintain security guarantees as software evolves, learned systems grow more complex, execution evidence remains incomplete, and AI-generated analysis results must be validated before they can be trusted. Across these settings, the central challenge is to determine **what remains guaranteed, which assumptions support it, and what must be revalidated or enforced when those assumptions change**. I pursue this agenda through four connected directions.

**Incremental validation for evolving software and systems.** Building on DataGuard and Uriah, I will develop

incremental security analyses that reuse previously established safety properties as software evolves, rather than reanalyzing and reprotecting the entire system after each change. When code, libraries, compiler transformations, or language boundaries change, the analysis will identify which objects, interfaces, and security assumptions are affected, invalidate only the corresponding guarantees, and concentrate new analysis or enforcement on those dependencies. I will apply this principle across heterogeneous and evolving software ecosystems, including multi-language systems, changing libraries and dependencies, compiler- and binary-level transformations, and cross-layer mechanisms involving verifiers, runtimes, speculation, and compartmentalization. The goal is to make strong security guarantees incrementally maintainable, so that the cost of preserving security scales with what changed rather than with the size of the entire system.

**Deployable guarantees for AI-enabled systems.** In light of my work on PolySafe and the DARPA VeriPro program, I will develop architecture-aware security-envelope synthesis for recurrent, convolutional, and attention-based models. For models that are too large for direct verification, I will develop verification-preserving simplification and distillation that bounds the deviation between the verified surrogate and deployed model, allowing guarantees to transfer soundly. I will couple these guarantees with runtime monitoring and recertification so that deployment can detect when certified conditions no longer hold and adapt accordingly. The goal is to synthesize and maintain enforceable guarantees as model architectures and deployment conditions evolve.

**Trustworthy security analysis under incomplete evidence.** I will develop methods that turn incomplete evidence into actionable security decisions. For fuzz testing, residual-risk estimates will guide when to stop, where to allocate additional effort, and how much uncertainty remains when no counterexample is found. For binary and other semantically incomplete software, I will recover only the security-relevant semantics needed for a target decision rather than reconstructing the entire program. I will further explore *evidence-guided forensic analysis* that combines partial artifacts such as crash traces, logs, and binary state, then directs additional analysis toward the evidence most likely to resolve the remaining uncertainty. The goal is to make explicit what is established, what remains unresolved, and where additional analysis should be directed.

**Validated generative reasoning for security analysis.** Generative models can search large spaces of program semantics, specifications, transformations, and candidate solutions that are difficult to explore with traditional analysis alone, but their outputs cannot be trusted directly. For example, an LLM may infer the likely semantics of an unknown binary function or propose an invariant for its state transitions; static analysis, symbolic reasoning, or execution can then test whether that hypothesis actually holds. I will build such generate-and-validate analyses so that generation expands the search space while independent validation determines which results can be trusted.

Over the longer term, I envision these directions converging toward **continuous assurance for evolving systems and software**. Rather than certifying a fixed artifact or configuration once, validation should continuously track what remains guaranteed, which assumptions still hold, and what new uncertainty emerges as the system changes. My long-term goal is to make validation-driven approach a practical foundation for preserving meaningful security guarantees as software, learned components, and deployment environments evolve.

**Open and reproducible research.** I treat open-source artifacts and reproducibility as part of the research process rather than as post-publication add-ons. I would like to highlight that although DataGuard and Uriah were developed separately, I later unified them into a memory-safety validation framework [42] and helped modernize the framework for newer LLVM versions [43]. I view maintaining usable research infrastructure, even when it does not naturally produce a separate publication, as an important contribution to reproducibility and follow-on research, and I will continue to make the artifacts from my future work openly available and maintainable whenever possible.

Looking forward, my research will build validation systems that retain meaningful guarantees despite evolving systems, software, learned behavior, and incomplete evidence. Across these settings, validation identifies what a guarantee depends on, establishes what can be rigorously supported, and makes residual uncertainty explicit when complete reasoning is infeasible. My long-term goal is to make **validation a practical foundation for deployable security**: rigorous where sound reasoning is possible, explicit about uncertainty where it is not, and clear about the assumptions on which each guarantee depends.

## References


[1] Laszlo Szekeres, Mathias Payer, Tao Wei, and Dawn Song. SoK: Eternal war in memory. In *2013 IEEE Symposium on Security and Privacy*, pages 48–62, San Francisco, CA, USA, May 2013. IEEE.

[2] Emanuel Q. Vintila, Philipp Zieris, and Julian Horsch. Evaluating the effectiveness of memory safety sanitizers. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 774–792, San Francisco, CA, USA, May 2025. IEEE.

[3] JongHyup Lee, Thanassis Avgerinos, and David Brumley. TIE: Principled reverse engineering of types in binary programs. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, February 2011. Internet Society.

[4] Kexin Pei, Jonas Guan, David Williams-King, Junfeng Yang, and Suman Jana. XDA: Accurate, robust disassembly with transfer learning. In *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS)*, Virtual Event, February 2021. Internet Society.

[5] Ali Ranjbar, Tianchang Yang, Kai Tu, Saaman Khalilollahi, and Syed Rafiul Hussain. Stateful analysis and fuzzing of commercial baseband firmware. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1120–1139. IEEE, 2025.

[6] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, March 2016.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, May 2017.

[8] Marcel Böhme, Danushka Liyanage, and Valentin Wüstholz. Estimating residual risk in greybox fuzzing. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 230–241. ACM, August 2021.

[9] Kaiming Huang. *Practical Comprehensive Memory Safety*. Ph.D. Dissertation, The Pennsylvania State University, University Park, PA, USA, 2024.

[10] Kaiming Huang, Yongzhe Huang, Mathias Payer, Zhiyun Qian, Jack Sampson, Gang Tan, and Trent Jaeger. The taming of the stack: Isolating stack data from memory errors. In *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, April 2022. Internet Society.

[11] Kaiming Huang, Mathias Payer, Zhiyun Qian, Jack Sampson, Gang Tan, and Trent Jaeger. Top of the heap: Efficient memory error protection of safe heap objects. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1330–1344, Salt Lake City, UT, USA, October 2024. ACM.

[12] Kaiming Huang, Mathias Payer, Zhiyun Qian, Jack Sampson, Gang Tan, and Trent Jaeger. SoK: Challenges and paths toward memory safety for eBPF. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 848–866, San Francisco, CA, USA, May 2025. IEEE.

[13] Rahul George, Mingming Chen, Kaiming Huang, Zhiyun Qian, Thomas La Porta, and Trent Jaeger. OPTISAN: Using multiple spatial error defenses to optimize stack memory protection within a budget. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 7195–7212, Philadelphia, PA, USA, August 2024. USENIX Association.

[14] Tianrou Xia, Kaiming Huang, Dongyeon Yu, Yuseok Jeon, Jie Zhou, Dinghao Wu, and Taegyu Kim. Litersan: Lightweight memory safety via rust-specific program analysis and selective instrumentation. Submitted to the Network and Distributed System Security Symposium (NDSS 2027), 2027.

[15] Sixuan Dang, Quan Zhou, Kaiming Huang, Jianan Zhu, Anitha Gollamudi, Trent Jaeger, and Danfeng Zhang. DualGuard: Comprehensive and efficient memory safety across sequential and speculative execution. Network and Distributed System Security Symposium (NDSS 2027), 2027.

[16] Yongzhe Huang, Vikram Narayanan, David Detweiler, Kaiming Huang, Gang Tan, Trent Jaeger, and Anton Burtsev. KSplit: Automating device driver isolation. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 613–631, Carlsbad, CA, USA, July 2022. USENIX Association.

[17] Yongzhe Huang, Kaiming Huang, Matthew Ennis, Vikram Narayanan, Anton Burtsev, Trent Jaeger, and Gang Tan. Beyond driver isolation: Triaging threats against driver isolation. In *Proceedings of the 2025 Annual Computer Security Applications Conference (ACSAC 2025)*, pages 245–261, Honolulu, HI, USA, December 2025.

[18] Yongzhe Huang, Kaiming Huang, Dongrui Zeng, Derrick McKee, Nathan Burow, Hamed Okhravi, Trent Jaeger, and Gang Tan. Interspec: Static analysis guided policy inference and enforcement for interface safety in compartmentalized applications. Manuscript, 2026.

[19] Kaiming Huang, Peng Wu, Mahdi Imani, Tian Lan, and Gang Tan. Validating safety guarantees of lstm models in mr context. In *Proceedings of the 2025 International Symposium on Theory, Algorithmic Foundations, and Protocol Design*

*for Mobile Networks and Mobile Computing (MobiHoc '25)*, pages 507–508, Houston, TX, USA, 2025. Association for Computing Machinery.

[20] Peng Wu, Nasim Ahmed, Abhiram Sarma, Kaiming Huang, Rifatul Islam, Bin Li, Tian Lan, Gang Tan, and Mahdi Imani. Probabilistic verification of cybersickness in virtual reality through bayesian networks. In *2025 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 782–792. IEEE, 2025.

[21] Peng Wu, Kaiming Huang, Nasim Ahmed, Md Mahedi Hassan, Mingyu Zhu, Peyton S. Bailey, Jessyca L. Derby, John Kwasinski, JoAnn Archer, Rifatul Islam, Bin Li, Tian Lan, Gang Tan, and Mahdi Imani. Predicting, verifying, and characterizing cybersickness over extended mixed-reality exposure in tactical training. Submitted to the IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR 2027), 2027.

[22] Kaiming Huang, Jialun Zhang, Shu Hong, Peng Wu, Nasim Ahmed, Jianan Jiang, Rifatul Islam, Mahdi Imani, Tian Lan, Bin Li, and Gang Tan. PolySafe: Synthesizing enforceable security envelopes for recurrent neural networks. Submitted to the Network and Distributed System Security Symposium (NDSS 2027), 2027.

[23] Nasim Ahmed, Peng Wu, Kaiming Huang, Sungchul Jung, Hansol Rheem, Gang Tan, Mahdi Imani, and Rifatul Islam. Human task performance and associated internal states in extended reality: A systematic review of cognitive, psychophysiological, and physiological dimensions. *Frontiers in Virtual Reality*, 6:1589256, 2025.

[24] Nasim Ahmed, Md Mahedi Hassan, Peng Wu, Kaiming Huang, Mingyu Zhu, JoAnn Archer, John Kwasinski, Peyton S. Bailey, Jessyca L. Derby, Tian Lan, Bin Li, Gang Tan, Mahdi Imani, and Rifatul Islam. Believe it. engage it.: A mixed reality dataset of engagement, cognitive load, physical load, cybersickness, task performance under cognitive attack. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2026.

[25] Nasim Ahmed, Peng Wu, Kaiming Huang, Tian Lan, Bin Li, Gang Tan, Mahdi Imani, and Rifatul Islam. One size does not fit all: Personalized cybersickness forecasting with asymmetric few-shot meta-learning. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2026.

[26] Nasim Ahmed, Peng Wu, Kaiming Huang, Bin Li, Tian Lan, Gang Tan, Mahdi Imani, and Rifatul Islam. Not everyone gets sick the same way: Personalized cybersickness prediction via bayesian networks with individual susceptibility modeling. *IEEE Transactions on Visualization and Computer Graphics*, 2026. Presented at the IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2026).

[27] Peng Wu, Kaiming Huang, Mingyu Zhu, Zifan Zhou, Jianan Jiang, Nasim Ahmed, Md Mahedi Hassan, Shu Hong, Bin Li, Rifatul Islam, Tian Lan, Gang Tan, and Mahdi Imani. Robust engagement estimation in mixed reality under cognitive attack. *IEEE Transactions on Visualization and Computer Graphics*, 2026. Presented at the IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2026).

[28] Peng Wu, Mingyu Zhu, Zifan Zhou, Jianan Jiang, Haowei Li, Juaren Steiger, Nasim Ahmed, Kaiming Huang, Gang Tan, Tian Lan, Rifatul Islam, Jonathan Dodge, and Mahdi Imani. Lag you can feel: Within-person links between latency, performance, and trust in mixed reality. In *2nd Workshop on Enhancing Security, Privacy, and Trust in Extended Reality Systems (XR Security), co-located with ACM MobiCom*, 2026.

[29] Shu Hong, Kaiming Huang, Mingyu Zhu, Zifan Zhou, Jianan Jiang, Nasim Ahmed, Peng Wu, Bin Li, Rifatul Islam, Mahdi Imani, Gang Tan, and Tian Lan. Certified gaze monitoring for cognitive attacks in mixed reality. Submitted to the IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR 2027), 2027.

[30] Peng Wu, Nasim Ahmed, Kaiming Huang, Md Mahedi Hassan, Mingyu Zhu, Peyton S. Bailey, Jessyca L. Derby, John Kwasinski, JoAnn Archer, Rifatul Islam, Bin Li, Tian Lan, Gang Tan, and Mahdi Imani. Forecasting post-exposure cybersickness symptom burden and recovery in mixed reality. Submitted to the IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR 2027), 2027.

[31] Srivatsa Kamballa, Kaiming Huang, Rahul R. Shevade, Yannic Noller, Gang Tan, and Saeid Tizpaz-Niari. A bayesian approach to estimating residual risk of fuzzing. Submitted to the International Conference on Software Engineering (ICSE 2027), 2027.

[32] Kaiming Huang, Jack Sampson, and Trent Jaeger. Assessing the impact of efficiently protecting ten million stack objects from memory errors comprehensively. In *2023 IEEE Secure Development Conference (SecDev)*, pages 67–74, Atlanta, GA, USA, October 2023. IEEE.

[33] Derrick McKee, Yianni Giannaris, Carolina Ortega Perez, Howard Shrobe, Mathias Payer, Hamed Okhravi, and Nathan Burow. Preventing kernel hacks with HAKC. In *Proceedings of the 29th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, California, USA, April 2022. Internet Society.

[34] Derrick McKee, Kaiming Huang, Nathan Burow, Trent Jaeger, Hamed Okhravi, and Gang Tan. HAKC–MSV: Memory Safety Validation. `https://github.com/HAKC-MSV`, 2026. Open-source implementation.

[35] Derrick McKee, Kaiming Huang, Nathan Burow, Trent Jaeger, Hamed Okhravi, and Gang Tan. HAKC–MSV: Hardware-Assisted Kernel Compartmentalization and Memory Safety Verification. https://hakc-msv.github.io/, 2026. Project website.

[36] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. $AI^2$: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, San Francisco, CA, USA, May 2018. IEEE.

[37] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1599–1614, Baltimore, MD, USA, August 2018. USENIX Association.

[38] Ching-Yun Ko, Zhaoyang Lyu, Lily Weng, Luca Daniel, Ngai Wong, and Dahua Lin. POPQORN: Quantifying robustness of recurrent neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 3468–3477. PMLR, June 2019.

[39] Wonryong Ryou, Jiayu Chen, Mislav Balunović, Gagandeep Singh, Andrei Dan, and Martin Vechev. Scalable polyhedral verification of recurrent neural networks. In Alexandra Silva and K. Rustan M. Leino, editors, *Computer Aided Verification (CAV)*, volume 12759 of *Lecture Notes in Computer Science*, pages 225–248, Cham, 2021. Springer.

[40] Zhouxing Shi, Qirui Jin, Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. Neural network verification with branch-and-bound for general nonlinearities. In Arie Gurfinkel and Marijn Heule, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 15696 of *Lecture Notes in Computer Science*, pages 315–335, Cham, 2025. Springer.

[41] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):41:1–41:30, January 2019.

[42] Kaiming Huang and collaborators. Unified memory safety validation. https://github.com/Lightninghkm/Unified-Memory-Safety-Validation. Open-source software.

[43] Kaiming Huang and collaborators. Msv: Unified memory safety validation with modern llvm support. https://github.com/HAKC-MSV/MSV. Open-source software.