Horizon 2020 Program (2014-2020)

Cybersecurity, Trustworthy ICT Research & Innovation Actions Security-by-design for end-to-end security
ICT 32-2014

Secure Hardware-Software Architectures for
Robust Computing Systems [†]

Deliverable D6.4: Dissemination report, year 2

Abstract: This report summarizes the dissemination activities carried out by the SHARCS project in year two. Specifically, in the following pages we list the papers presented by the consortium in conferences as well as the presentations made at various events and forums, related to the project. Additional coverage of the project through other dissemination channels is also presented in this document, including releases in the popular press, references to the project and established liaisons with related EU projects. During the second year of SHARCS, the consortium published a total of 24 peer-reviewed papers, plus 8 presentations, talks and seminars. A notable success is that one of the SHARCS publications won the Outstanding Paper Award at CODASPY 2016. Also, another paper was awarded Best Paper Session at DSN 2016. Furthermore, one other publication won the Pwnie Award at Black Hat USA 2016. Moreover, the informational leaflet was updated and one poster was prepared. These appear in the Appendices of this deliverable.

Contractual Date of Delivery	Month 24
Actual Date of Delivery	Month 25
Deliverable Dissemination Level	Public
Editors	Marinos Tsantekidis
Contributors	All SHARCS partners
Quality Assurance	George Christou, Christos Strydis

[†] The research leading to these results has received funding from the European Union Horizon 2020 Program (2014-2020) under grant agreement n° 644571.

The SHARCS Consortium

Foundation for Research and Technology – Hellas	Coordinator	Greece
Vrije Universiteit Amsterdam	Principal Contractor	The Netherlands
Chalmers Tekniska Högskola	Principal Contractor	Sweden
Technische Universität Braunschweig	Principal Contractor	Germany
Neurasmus BV	Principal Contractor	The Netherlands
OnApp Limited	Principal Contractor	United Kingdom
IBM - Science and Technology LTD	Principal Contractor	Israel
Elektrobit Automotive GMBH	Principal Contractor	Germany

Document Revisions & Quality Assurance

Internal Reviewers

1. George Christou,
2. Christos Strydis.

Revisions

Ver.	Date	By	Overview
0.1	25/10/2016	Marinos Tsantekidis	Abstract and main text
0.2	15/11/2016	Cristiano Giuffrida	VUA publications and courses
0.3	24/11/2016	Marinos Tsantekidis	Added dissemination activities
0.4	30/11/2016	Christos Strydis	NEU dissemination information added Dissemination
0.5	30/11/2016	Marinos Tsantekidis	Ported to LaTeX, added CSW photos
0.6	01/12/2016	Marinos Tsantekidis	Added keynote speak
0.7	28/12/2016	Marinos Tsantekidis	Addressed internal review comments
0.8	27/01/2017	Cristiano Giuffrida	VUA additions
0.9	31/01/2017	John Thomson	Final corrections
1.0	01/02/2017	Marinos Tsantekidis	Addressed review comments and released final version

Contents

1 Publications

SHARCS aims to disseminate the work carried out to many venues through a number of papers. In the second year, we had 24 publications with a variety of subjects including hardware modifications, several attacks to account for attack vectors not considered in the SHARCS models, compiler/linker extensions and policy access control. Several of these papers received important awards at their respective conferences and recognition from the community. They are described below, along with their abstracts.

1. N. Christoulakis, G. Christou, E. Athanasopoulos, S. Ioannidis, **"HCFI: Hardware-enforced Control-Flow Integrity"**, *Proceedings of the 6th ACM Conference on Data and Applications Security and Privacy (CODASPY),* 2016.

   This paper received the Outstanding Paper Award.

   Relevance to SHARCS: This work adds to the first model of the SHARCS framework, the "Clean-Slate Approach". Modifications are made to the hardware itself, in this case the processor, to offer extensions that enforce Control-Flow Integrity; thereby thwarting a number of attacks such as Return-Oriented Programming.

   Abstract: Control-flow hijacking is the principal method for code-reuse techniques like Return-oriented Programming (ROP) and Jump-oriented Programming (JOP). For defending against such attacks, the community has proposed Control-flow Integrity (CFI), a technique capable of preventing exploitation by verifying that every (indirect) control-flow transfer points to a legitimate address. Enabling CFI in real systems is not straightforward, since in many cases the actual Control-flow Graph (CFG) of a program can be only approximated. Even in the case that there is perfect knowledge of the CFG, ensuring that all return instructions will return to their actual call sites, without employing a shadow stack, is questionable. On the other hand, the community has expressed concerns related to significant overheads stemming from enabling a shadow stack. In this paper, we acknowledge the importance of a shadow stack for supporting and strengthening any CFI policy. In addition, we project that implementing a full-featured CFI enabled Instruction Set Architecture (ISA) in actual hardware with an in-chip secure memory can be efficiently carried out and the prototype experiences negligible overheads. For supporting our case, we implement HCFI by modifying a SPARC SoC and evaluate the prototype on an FPGA board by running all SPECInt benchmarks instrumented with a fine-grained CFI policy. The evaluation shows that HCFI can effectively protect applications from code-reuse attacks, while adding less than 1% runtime overhead.

2. E. Degkleri, A. A. Chariton, P. Ilia, P. Papadopoulos, E. P. Markatos, **"Leveraging DNS for timely SSL Certificate Revocation"**, *Proceedings of ACM-W Europe Celebration of Women in Computing (womENcourage),* 2016.

Relevance to SHARCS: This work is an enhancement in SSL-based communication. The performance results of this approach make it a suitable alternative for use cases which make use of SSL and have strict performance requirements.

Abstract: Trust in SSL-based communication on the Internet is provided by Certificate Authorities in the form of signed certificates. When an organization uses an SSL certificate, it protects user-sensitive information by encrypting all traffic between its servers and the user's web browser. Sadly, current web browser approaches to check the revocation status of a certificate, suffer from certain performance issues and privacy implications. To address these issues, we propose DCSP: a new low-latency approach that by leveraging the existing infrastructure of DNS, provides performant and accurate certificate revocation information. Our initial performance results show that DCSP has the potential to perform an order of magnitude faster than the current state-of-the-art alternatives.

3. I. Haller, J. Yuseok, H. Peng, M. Payer, C. Giuffrida, H. Bos, and E. van der Kouwe **"TypeSan: Practical Type Confusion Detection"**, *Proceedings of the ACM Conference on Computer and Communications Security (CCS),* 2016.

   Relevance to SHARCS: This work proposes extensions to the compiler to efficiently detect type-confusion bugs (a dangerous class of memory error vulnerabilities). This is a software-only defense relevant for the third model of the SHARCS framework.

   Abstract: The low-level C++ programming language is ubiquitously used for its modularity and performance. Typecasting is a fundamental concept in C++ (and object-oriented programming in general) to convert a pointer from one object type into another. However, downcasting (converting a base class pointer to a derived class pointer) has critical security implications due to potentially different object memory layouts. Due to missing type safety in C++, a downcasted pointer can violate a programmer's intended pointer semantics, allowing an attacker to corrupt the underlying memory in a type-unsafe fashion. This vulnerability class is receiving increasing attention and is known as type confusion (or 'badcasting'). Several existing approaches detect different forms of type confusion, but these solutions are severely limited due to both high run-time performance overhead and low detection coverage. This paper presents TypeSan, a practical type-confusion detector which provides both low run-time overhead and high detection coverage. Despite improving the coverage of state-of-the-art techniques, TypeSan significantly reduces the type-confusion detection overhead compared to other solutions. TypeSan relies on an efficient per-object metadata storage service based on a compact memory shadowing scheme. Our scheme treats all the memory objects (i.e., globals, stack, heap) uniformly to eliminate extra checks on the fast path and relies on a variable compression ratio to minimize run-time performance and memory overhead. Our experimental results confirm that TypeSan is practical, even when explicitly checking almost all the relevant typecasts in a given C++ program. Compared to the state of the art, TypeSan yields orders of magnitude higher coverage at 4 – 10 times lower performance overhead on SPEC and 2 times on Firefox. As a result, our solution offers superior protection and is suitable for deployment in production software. Moreover, our highly efficient metadata storage back-end is potentially useful for other defenses that require memory object tracking.

4. V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, **"Drammer: Deterministic Rowhammer Attacks on Mobile Platforms"**, *Proceedings of CCS,* 2016.

   This publication received extensive media coverage and prompted Google to deploy proposed countermeasures on Android.

   Relevance to SHARCS: This work demonstrates that hardware (Rowhammer) attacks can be used to compromise widespread mobile devices in practical settings. This is

relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

Abstract: Recent work shows that the Rowhammer hardware bug can be used to craft powerful attacks and completely subvert a system. However, existing efforts either describe probabilistic (and thus unreliable) attacks or rely on special (and often unavailable) memory management features to place victim objects in vulnerable physical memory locations. Moreover, prior work only targets x86 and researchers have openly wondered whether Rowhammer attacks on other architectures, such as ARM, are even possible. We show that deterministic Rowhammer attacks are feasible on commodity mobile platforms and that they cannot be mitigated by current defenses. Rather than assuming special memory management features, our attack, Drammer, solely relies on the predictable memory reuse patterns of standard physical memory allocators. We implement Drammer on Android/ARM, demonstrating the practicability of our attack, but also discuss a generalization of our approach to other Linux-based platforms. Furthermore, we show that traditional x86-based Rowhammer exploitation techniques no longer work on mobile platforms and address the resulting challenges towards practical mobile Rowhammer attacks. To support our claims, we present the first Rowhammer based Android root exploit relying on no software vulnerability, and requiring no user permissions. In addition, we present an analysis of several popular smartphones and find that many of them are susceptible to our Drammer attack. We conclude by discussing potential mitigation strategies and urging our community to address the concrete threat of faulty DRAM chips in widespread commodity platforms.

5. T. Hruby, C. Giuffrida, L. Sambuc, H. Bos, and A. S. Tanenbaum, **"A NEaT Design for Reliable and Scalable Network Stacks"**, *Proceedings of CoNext,* 2016.

   Relevance to SHARCS: This work proposes a new replicated network stack design for scalability, reliability, and security. The replication allows network requests to be served by randomized network stack variants, mitigating remote exploitation attempts. The latter is a software-only defense relevant for the third model of the SHARCS framework.

   Abstract: Operating systems provide a wide range of services, which are crucial for the increasingly high reliability and scalability demands of modern applications. Providing both reliability and scalability at the same time is hard. Commodity OS architectures simply lack the design abstractions to do so for demanding core OS services such as the network stack. For reliability and scalability guarantees, they rely almost exclusively on ensuring a high-quality implementation, rather than a reliable and scalable design. This results in complex error recovery paths and hard-to-maintain synchronization code. We demonstrate that a simple and structured design that strictly adheres to two principles, isolation and partitioning, can yield reliable and scalable network stacks. We present NEaT, a system which partitions the stack across isolated process replicas handling independent requests. Our design principles intelligently partition the state to minimize the impact of failures (offering strong recovery guarantees) and to scale comparably to Linux without exposing the implementation to common pitfalls such as synchronization errors, poor locality, and false sharing.

6. P. Sarbinowski, V. P. Kemerlis, C. Giuffrida, and E. Athanasopoulos, **"VTPin: Practical Table Hijacking Protection for Binaries"**, *Proceedings of ACSAC,* 2016.

   Relevance to SHARCS: This work proposes extensions to the dynamic linker to efficiently mitigate control-flow diversion attacks based on vtable hijacking and use-after-free vulnerabilities. This is a software-only defense relevant for the third model of the SHARCS framework.

   Abstract:VTable hijacking has lately been promoted to the defacto technique for exploiting C++ applications and in particular web browsers. VTables, however, can be manipulated without necessarily corrupting memory, simply by leveraging use-after-free bugs. In fact, in the recent Pwn2Own competitions all major

web browsers were compromised with exploits that employed (among others) use-after-free vulnerabilities and VTable hijacking. In this paper, we propose VTPin: a system to protect against VTable hijacking, via use-after-free vulnerabilities, in large C++ binaries that cannot be re-compiled or re-written. The main idea behind VTPin is to pin all the freed VTable pointers on a safe VTable under VTPin's control. Specifically, for every object deallocation, VTPin deallocates all space allocated, but preserves and updates the VTable pointer with the address of the safe VTable. Hence, any dereferenced dangling pointer can only invoke a method provided by VTPin's safe object. Subsequently, all virtual-method calls due to dangling pointers are not simply neutralized, but they can be logged, tracked, and patched. Compared to other solutions that defend against VTable hijacking, VTPin exhibits certain characteristics that make it suitable for practical and instant deployment in production software. First, VTPin protects binaries, directly and transparently, without requiring source compilation or binary rewriting. Second, VTPin is not an allocator replacement, and thus it does not interfere with the allocation strategies and policies of the protected program; it intervenes in the deallocation process only when a virtual object is to be freed for preserving the VTable pointer. Third, VTPin is fast; Mozilla Firefox, protected with VTPin, experiences an average overhead of 1% - 4.1% when running popular browser benchmarks.

7. A. Miraglia, D. Vogt, H. Bos, A. S. Tanenbaum, and C. Giuffrida, **"Peeking into the Past: Efficient Checkpoint-assisted Time-traveling Debugging"**, *Proceedings of ISSRE,* 2016.

   Relevance to SHARCS: This work proposes a memory checkpointing framework and describes its application to software debugging. More generally, the memory checkpointing framework can serve as a building block for a general-purpose exploit recovery solution. The latter is a software-only defense relevant for the third model of the SHARCS framework.

   Abstract: Debugging long-lived latent software bugs that manifest themselves only long after their introduction in the system, is hard. Even state-of-the-art record/replay debugging techniques are of limited use to identify the root cause of long-lived latent bugs in general and event-driven bugs in particular. We propose DeLorean, a new end-to-end solution for time-travelling debugging based on fast memory checkpointing. Our design trades off replay guarantees with efficient support for history-aware debug queries (or time-travelling introspection) and provides novel analysis tools to diagnose event-driven latent software bugs. DeLorean imposes low run-time performance and memory overhead while preserving in memory as much history information as possible by deduplicating and/or compressing the collected data. We evaluate DeLorean by extensive experimentation, exploring the performance-memory tradeoffs in different configurations and comparing our results against state-of-the-art solutions. We show that DeLorean can efficiently support high frequency checkpoints and store millions of them in memory

8. E. Goktas, A. Oikonomopoulos, R. Gawlik, B. Kollenda, E. Athanasopoulos, G. Portokalidis, C. Giuffrida, and Bos, Herbert, **"Bypassing Clang's SafeStack for Fun and Profit"**, *Proceedings of Black Hat Europe,* 2016.

   Relevance to SHARCS: This work demonstrates that side-channel attacks can be used to bypass state-of-the-art shadow stack defenses. This is relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

   Abstract: SafeStack, a new compiler feature currently only available in clang and underway for GCC, protects return addresses on the stack from being overwritten through memory vulnerabilities. SafeStack (-fsanitize=safe-stack) is intended to replace the stack cookies (-fstack-protector). It separates the data and the return addresses on the original stack, and puts the former in the unsafe stack and the latter in the safe stack. We investigate the implementation of the safe stack to see if there are still ways to get to it and overwrite the return addresses. In this work, we show implementation issues that allow an attacker

to get to the safe stack. In addition, we demonstrate two new fundamental strategies to efficiently find the safe stack, namely through thread spraying and allocation oracles. Thread spraying is a technique to force the application to spawn many safe stacks and to reduce the entropy of the safe stacks significantly. With allocation oracles we can determine the sizes of the unallocated holes in the address space and as such the distance from the known regions to the hidden regions.

9. K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, **"Flip Feng Shui: Breaking the VM's Isolation"**, *Proceedings of Black Hat Europe,* 2016.

   Relevance to SHARCS: This work demonstrates that hardware (Rowhammer) attacks can be used to compromise cloud systems with memory deduplication enabled. This is relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

   Abstract: We show how an attacker virtual machine (VM) can induce Rowhammer bit flips over memory used by another VM on the same host even under strong hardware-enforced memory isolation in a "fully controlled way". In many cloud settings, memory deduplication is used to reduce the memory footprint of VMs significantly by keeping a single copy of multiple memory pages with the same contents. The memory deduplication process scans the memory periodically to find memory pages with the same contents, then keeps one copy in the physical memory (i.e., the primary page) and releases the copies to the system. We show that by guessing the contents of a target page in a victim VM, an attacker VM can easily control the primary page, or in other words, the location of the victim's memory page on physical memory. By placing the victim page on a physical memory location with the right vulnerable bit offset, determined in the first stage of our exploit, we can perform a reliable and deterministic Rowhammer across VMs. We used this new technique, named 'flip feng shui', to corrupt the page cache of a victim VM hosting RSA public keys. We exemplify end-to-end attacks (a) breaking OpenSSH public-key authentication, thereby allowing remote OpenSSH access using a newly generated private key, and (b) forging GPG signatures from trusted keys, thereby compromising the Ubuntu/Debian updating mechanism, all without relying on any software vulnerability. Unlike other Rowhammer-based cryptographic fault attacks, ours is quite practical: it does not make any assumption on the environment nor requires the knowledge of the CPU's memory addressing function. We discuss practical defenses against flip feng shui attacks at the end.

10. E. Bosman, K. Razavi, H. Bos, C. Giuffrida, **"Over the Edge: Silently Owning Windows 10's Secure Browser"**, *Proceedings of Black Hat USA,* 2016.

    Relevance to SHARCS: This work demonstrates that side channels and hardware (Rowhammer) attacks can be used to compromise browsers on systems with memory deduplication enabled. This is relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

    Abstract: Memory deduplication, a well-known technique to reduce the memory footprint across virtual machines, is now also a default-on feature inside the Windows 10 operating system. Deduplication maps multiple identical copies of a physical page onto a single shared copy with copy-on-write semantics. As a result, a write to such a shared page triggers a page fault and is thus measurably slower than a write to a normal page. We leverage this side channel to build a weird machine and read arbitrary data in the system from the browser. By controlling the alignment and reuse of data in memory, we perform a byte-by-byte disclosure of high-entropy sensitive data, such as 64-bit code pointers randomized by ASLR. Next, even without control over data alignment or reuse, we show how to disclose randomized 64-bit heap pointers using a novel birthday attack. To show these attack primitives are practical, we have built an end-to-end JavaScript-based exploit against the new Microsoft Edge browser, in absence of software vulnerabilities and with all defenses turned on. Our exploit combines our deduplication-based primitives with a reliable Rowhammer attack to gain arbitrary memory read and write access in the browser.

11. A. Oikonomopoulos, C. Giuffrida, S. Rawat, H. Bos, **"Binary Rejuvenation: Applications and Challenges"**, *IEEE Security & Privacy Magazine. 14 (68-71),* 2016.

    Relevance to SHARCS: This work proposes a binary analysis and rewriting framework to rejuvenate legacy binaries and improve their security. This is a software-only defense framework relevant for the third model of the SHARCS framework.

    Abstract: Software engineers have long performed source code rejuvenation-rewriting obsolete or outdated programming idioms to modern counterparts. Taking inspiration from this practice, we suggest applying rejuvenation to the results of the compilation process. That is, we propose updating selected binary files, or binary rejuvenation.

12. I. Haller, E. van der Kouwe, C. Giuffrida, H. Bos, **"METAlloc: Efficient and Comprehensive Metadata Management for Software Security Hardening"**, *Proceedings of EuroSec,* 2016.

    Relevance to SHARCS: This work proposes an efficient metadata tracking framework to implement generic security hardening techniques against memory error vulnerabilities. This is a software-only defense framework relevant for the third model of the SHARCS framework.

    Abstract: Many systems software security hardening solutions rely on the ability to look up metadata for individual memory objects during the execution, but state-of-the-art metadata management schemes incur significant lookup time or allocation-time overheads and are unable to handle different memory objects (i.e., stack, heap, and global) in a comprehensive and uniform manner. We present METAlloc, a new memory metadata management scheme which addresses all the key limitations of existing solutions. Our design relies on a compact memory shadowing scheme empowered by an alignment-based object allocation strategy. METAlloc's allocation strategy ensures that all the memory objects within a page share the same alignment class and each object is always allocated to use the largest alignment class possible. This strategy provides a fast memory-to-metadata mapping, while minimizing metadata size and reducing memory fragmentation. We implemented and evaluated METAlloc on Linux and show that METAlloc (de)allocations incur just 3.6% run-time performance overhead, paving the way for practical software security hardening in real-world deployment scenarios.

13. E. Bosman, K. Razavi, H. Bos, C. Giuffrida, **"Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector"**, *Proceedings of S&P,* 2016.

    This publication won the Pwnie Award for most innovative research at Black Hat USA, 2016. It also received extensive media coverage and prompted Microsoft to disable memory deduplication.

    Relevance to SHARCS: This work demonstrates that side channels and hardware (Rowhammer) attacks can be used to compromise browsers on systems with memory deduplication enabled. This is relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

    Abstract: Memory deduplication, a well-known technique to reduce the memory footprint across virtual machines, is now also a default-on feature inside the Windows 8.1 and Windows 10 operating systems. Deduplication maps multiple identical copies of a physical page onto a single shared copy with copy-on-write semantics. As a result, a write to such a shared page triggers a page fault and is thus measurably slower than a write to a normal page. Prior work has shown that an attacker able to craft pages on the target system can use this timing difference as a simple single-bit side channel to discover that certain pages exist in the system. In this paper, we demonstrate that the deduplication side channel is much more powerful than previously assumed, potentially providing an attacker with a weird machine to read

arbitrary data in the system. We first show that an attacker controlling the alignment and reuse of data in memory is able to perform byte-by-byte disclosure of sensitive data (such as randomized 64 bit pointers). Next, even without control over data alignment or reuse, we show that an attacker can still disclose high-entropy randomized pointers using a birthday attack. To show these primitives are practical, we present an end-to-end JavaScript-based attack against the new Microsoft Edge browser, in absence of software bugs and with all defenses turned on. Our attack combines our deduplication-based primitives with a reliable Rowhammer exploit to gain arbitrary memory read and write access in the browser. We conclude by extending our JavaScript-based attack to cross-process system-wide exploitation (using the popular nginx web server as an example) and discussing mitigation strategies.

14. V. van der Veen, E. Goktas, M. Contag, A. Pawlowski, X. Chen, S. Rawat, H. Bos, T. Holz, E. Athanasopoulos, C. Giuffrida, **"A Tough call: Mitigating Advanced Code-Reuse Attacks At The Binary Level"**, *Proceedings of S&P,* 2016.

    Relevance to SHARCS: This work proposes a binary rewriting-based control-flow integrity solution that relies on type inference to enforce strong security invariants. This is a software-only defense relevant for the third model of the SHARCS framework.

    Abstract: Current binary-level Control-Flow Integrity (CFI) techniques are weak in determining the set of valid targets for indirect control flow transfers on the forward edge. In particular, the lack of source code forces existing techniques to resort to a conservative address-taken policy that over-approximates this set. In contrast, source-level solutions can accurately infer the targets of indirect callsites and thus detect malicious control-flow transfers more precisely. Given that source code is not always available, however, offering similar quality of protection at the binary level is important, but, unquestionably, more challenging than ever: recent work demonstrates powerful attacks, such as Counterfeit Object-oriented Programming (COOP), which made the community believe that protecting software against control-flow diversion attacks at the binary level is impossible. In this paper, we propose binary-level analysis techniques to significantly reduce the number of possible targets for indirect callsites. More specifically, we reconstruct a conservative approximation of target function prototypes by means of use-def analysis at possible callees. We then couple this with liveness analysis at each indirect callsite to derive a many-to-many relationship between callsites and target callees with a much higher precision compared to prior binary-level solutions. Experimental results on popular server programs and on SPEC CPU2006 show that TypeArmor, a prototype implementation of our approach, is efficient with a runtime overhead of less than 3%. Furthermore, we evaluate to what extent TypeArmor can mitigate COOP and other advanced attacks and show that our approach can significantly reduce the number of targets on the forward edge. Moreover, we show that TypeArmor breaks published COOP exploits, providing concrete evidence that strict binary-level CFI can still mitigate advanced attacks, despite the absence of source information or C++ semantics.

15. K. Bhat, D. Vogt, E. van der Kouwe, B. Gras, L. Sambuc, A.S. Tanenbaum, H. Bos, C. Giuffrida, **"OSIRIS: Efficient and Consistent Recovery of Compartmentalized Operating Systems"**, *Proceedings of DSN,* 2016.

    This paper was selected for the Best Paper session.

    Relevance to SHARCS: This work proposes compiler extensions to perform automated error (or exploit) recovery. The latter is a software-only defense relevant for the third model of the SHARCS framework.

    Abstract: Much research has gone into making operating systems more amenable to recovery and more resilient to crashes. Traditional solutions rely on partitioning the operating system (OS) to contain the effects of crashes within compartments and facilitate modular recovery. However, state dependencies among the compartments hinder recovery that is globally consistent. Such recovery typically requires expensive runtime dependency tracking which results in high performance overhead, high complexity and

a large Reliable Computing Base (RCB). We propose a lightweight strategy that limits recovery to cases where we can statically and conservatively prove that compartment recovery leads to a globally consistent state trading recoverable surface for a simpler and smaller RCB with lower performance overhead and maintenance cost. We present OSIRIS, a research OS design prototype that demonstrates efficient and consistent crash recovery. Our evaluation shows that OSIRIS effectively recovers from important classes of real-world software bugs with a modest RCB and low overheads.

16. K. Koning, H. Bos, C. Giuffrida, **"Secure and Efficient Multi-variant Execution Using Hardware-assisted Process Virtualization"**, *Proceedings of DSN,* 2016.

    Relevance to SHARCS: This work proposes an efficient multi-variant execution system based on hardware virtualization. This is a commodity hardware-supported defense relevant for the second model of the SHARCS framework.

    Abstract: Memory error exploits rank among the most serious security threats. Of the plethora of memory error containment solutions proposed over the years, most have proven to be too weak in practice. Multi-Variant eXecution (MVX) solutions can potentially detect arbitrary memory error exploits via divergent behavior observed in diversified program variants running in parallel. However, none have found practical applicability in security due to their non-trivial performance limitations. In this paper, we present MvArmor, an MVX system that uses hardware-assisted process virtualization to monitor variants for divergent behavior in an efficient yet secure way. To provide comprehensive protection against memory error exploits, MvArmor relies on a new MVX-aware variant generation strategy. The system supports user-configurable security policies to tune the performance-security trade-off. Our analysis shows that MvArmor can counter many classes of modern attacks at the cost of modest performance overhead, even with conservative detection policies.

17. E. Goktas, R. Gawlik, B. Kollenda, E. Athanasopoulos, G. Portokalidis, C. Giuffrida, H. Bos, **"Undermining Information Hiding (And What to do About it)"**, *Proceedings of USENIX Security,* 2016.

    This publication prompted Mozilla to deploy targeted proposed countermeasures against thread spraying attacks.

    Relevance to SHARCS: This work demonstrates that side-channel attacks can be used to bypass state-of-the-art shadow stack defenses. This is relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

    Abstract: In the absence of hardware-supported segmentation, many state-of-the-art defenses resort to hiding sensitive information at a random location in a very large address space. This paper argues that information hiding is a weak isolation model and shows that attackers can find hidden information, such as CPI's SafeStack, in seconds by means of thread spraying. Thread spraying is a novel attack technique which forces the victim program to allocate many hidden areas. As a result, the attacker has a much better chance to locate these areas and compromise the defense. We demonstrate the technique by means of attacks on Firefox, Chrome, and MySQL. In addition, we found that it is hard to remove all sensitive information (such as pointers to the hidden region) from a program and show how residual sensitive information allows attackers to bypass defenses completely. We also show how we can harden information hiding techniques by means of an Authenticating Page Mapper (APM) which builds on a user-level page-fault handler to authenticate arbitrary memory reads/writes in the virtual address space. APM bootstraps protected applications with a minimum-sized safe area. Every time the program accesses this area, APM authenticates the access operation, and, if legitimate, expands the area on demand. We demonstrate that APM hardens information hiding significantly while increasing the overhead, on average, 0.3% on baseline SPEC CPU 2006, 0.0% on SPEC with SafeStack and 1.4% on SPEC with CPI.

18. K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, H. Bos, **"Flip Feng Shui: Hammering a Needle in the Software Stack"**, *Proceedings of USENIX Security,* 2016.

    This publication had extensive media coverage, prompted GnuPG to strengthen their security checks, and many cloud providers to disable memory deduplication.

    Relevance to SHARCS: This work demonstrates that hardware (Rowhammer) attacks can be used to compromise cloud systems with memory deduplication enabled. This is relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

    Abstract: We introduce Flip Feng Shui (FFS), a new exploitation vector which allows an attacker to induce bit flips over arbitrary physical memory in a fully controlled way. FFS relies on hardware bugs to induce bit flips over memory and on the ability to surgically control the physical memory layout to corrupt attacker-targeted data anywhere in the software stack. We show FFS is possible today with very few constraints on the target data, by implementing an instance using the Rowhammer bug and memory deduplication (an OS feature widely deployed in production). Memory deduplication allows an attacker to reverse-map any physical page into a virtual page she owns as long as the page's contents are known. Rowhammer, in turn, allows an attacker to flip bits in controlled (initially unknown) locations in the target page. We show FFS is extremely powerful: a malicious VM in a practical cloud setting can gain unauthorized access to a co-hosted victim VM running OpenSSH. Using FFS, we exemplify end-to-end attacks breaking OpenSSH public-key authentication, and forging GPG signatures from trusted keys, thereby compromising the Ubuntu/Debian update mechanism. We conclude by discussing mitigations and future directions for FFS attacks.

19. A. Oikonomopoulos, E. Athanasopoulos, H. Bos, C. Giuffrida, **"Poking Holes in Information Hiding"**, *Proceedings of USENIX Security,* 2016.

    Relevance to SHARCS: This work demonstrates that side-channel attacks can be used to bypass state-of-the-art defenses that rely on information hiding. This is relevant to quantify the residual attack surface for vectors not considered in the SHARCS threat model.

    Abstract: ASLR is no longer a strong defense in itself, but it still serves as a foundation for sophisticated defenses that use randomization for pseudo-isolation. Crucially, these defenses hide sensitive information (such as shadow stacks and safe regions) at a random position in a very large address space. Previous attacks on randomization-based information hiding rely on complicated side channels and/or probing of the mapped memory regions. Assuming no weaknesses exist in the implementation of hidden regions, the attacks typically lead to many crashes or other visible side-effects. For this reason, many researchers still consider the pseudo-isolation offered by ASLR sufficiently strong in practice. We introduce powerful new primitives to show that this faith in ASLR-based information hiding is misplaced, and that attackers can break ASLR and find hidden regions on 32 bit and 64 bit Linux systems quickly with very few malicious inputs. Rather than building on memory accesses that probe the allocated memory areas, we determine the sizes of the unallocated holes in the address space by repeatedly allocating large chunks of memory. Given the sizes, an attacker can infer the location of the hidden region with few or no side-effects. We show that allocation oracles are pervasive and evaluate our primitives on real-world server applications.

20. D. Andriesse, X. Chen, V. van der Veen, A. Slowinska, H. Bos, **"An In-Depth Analysis of Disassembly on Full-Scale x86/x64 Binaries"**, *Proceedings of USENIX Security,* 2016.

    Relevance to SHARCS: This work presents an in-depth analysis of disassembly to improve our understanding of the properties of widespread x86 binaries. This is important to build practical software-only defenses for legacy binaries, relevant for the third model of the SHARCS framework.

Abstract: It is well-known that static disassembly is an unsolved problem, but how much of a problem is it in real software, for instance, for binary protection schemes? This work studies the accuracy of nine state-of-the-art disassemblers on 981 real-world compiler-generated binaries with a wide variety of properties. In contrast, prior work focuses on isolated corner cases; we show that this has led to a widespread and overly pessimistic view on the prevalence of complex constructs like inline data and overlapping code, leading reviewers and researchers to underestimate the potential of binary-based research. On the other hand, some constructs, such as function boundaries, are much harder to recover accurately than is reflected in the literature, which rarely discusses much needed error handling for these primitives. We study 30 papers recently published in six major security venues, and reveal a mismatch between expectations in the literature, and the actual capabilities of modern disassemblers. Our findings help improve future research by eliminating this mismatch.

21. L. Koromilas, G. Vasiliadis, E. Athanasopoulos, S. Ioannidis, **"GRIM: Leveraging GPUs for Kernel Integrity Monitoring"**, *Proceedings of International Symposium on Research in Attacks, Intrusions and Defenses (RAID),* 2016.

    Relevance to SHARCS: This work enhances the Operating System security against rootkits. It utilizes GPGPUs in order to offload the computation needed from CPUs.

    Abstract: Kernel rootkits can exploit an operating system and enable future accessibility and control, despite all recent advances in software protection. A promising defense mechanism against rootkits is Kernel Integrity Monitor (KIM) systems, which inspect the kernel text and data to discover any malicious changes. A KIM can be implemented either in software, using a hypervisor, or using extra hardware. The latter option is more attractive due to better performance and higher security, since the monitor is isolated from the potentially vulnerable host. To remain under the radar and avoid detection it is paramount for a rootkit to conceal its malicious activities. In order to detect self-hiding rootkits researchers have proposed snooping for inferring suspicious behaviour in kernel memory. This is accomplished by constantly monitoring all memory accesses on the bus and not the actual memory area where the kernel is mapped. In this paper, we present GRIM, an external memory monitor that is built on commodity, off-the-shelf, graphics hardware, and is able to verify OS kernel integrity at a speed that outperforms all so-far published snapshot-based systems. GRIM allows for checking eight thousand 64-bit values simultaneously at a 10 KHz snapshot frequency, which is sufficient to accurately detect a self-hiding loadable kernel module insertion. According to the state-of-the-art, this detection can only happen using a snoop-based monitor. GRIM does not only demonstrate that snapshot-based monitors can be significantly improved, but it additionally offers a fully programmable platform that can be instantly deployed without requiring any modifications to the host it protects. Notice that all snoopbased monitors require substantial changes at the microprocessor level.

22. R.M. Seepers, C. Strydis, J.H. Weber, Z. Erkin, and I. Sourdis, **"Secure Key-Exchange Protocol for Implants Using Heartbeats"**, *Proceedings of ACM International Conference on Computing Frontiers,* 2016.

    Relevance to SHARCS: This work complements our previous work on IPI-based, security-key generation by proposing a key-exchange protocol that is resistant to heart-beat mis-detections. It is lightweight, energy-efficient and enables practical implementations of IPI-based implant security.

    Abstract: The cardiac interpulse interval (IPI) has recently been proposed to facilitate key exchange for implantable medical devices (IMDs) using a patient's own heartbeats as a source of trust. While this form of key exchange holds promise for IMD security, its feasibility is not fully understood due to the simplified approaches found in related works. For example, previously proposed protocols have been designed without considering the limited randomness available per IPI, or have overlooked aspects pertinent to a realistic system, such as imperfect heartbeat detection or the energy overheads imposed on an IMD. In this

paper, we propose a new IPI-based key-exchange protocol and evaluate its use during medical emergencies. Our protocol employs fuzzy commitment to tolerate the expected disparity between IPIs obtained by an external reader and an IMD, as well as a novel way of tackling heartbeat misdetection through IPI classification. Using our protocol, the expected time for securely exchanging an 80-bit key with high probability $(1 - 10^{-6})$ is roughly one minute, while consuming only 88 μJ from an IMD.

23. A. Karapatis, R.M. Seepers, M. van Dongen, W.A. Serdijn, C. Strydis, **"Balancing Accuracy, Delay and Battery Autonomy for Pervasive Seizure Detection"**, *Proceedings of IEEE Engineering in Medicine and Biology Society (EMBC),* 2016.

    Relevance to SHARCS: This work improves on the seizure-prevention application that is the crux of the SHARCS implant use case. An in-depth analysis is performed of different design points and optimization parameters, leading to a departure from the current state of the art. Optimizations contribute not only to improved seizure detection but also to more energy-saving operation.

    Abstract: A promising alternative for treating absence seizures has emerged through closed-loop neurostimulation, which utilizes a wearable or implantable device to detect and subsequently suppress epileptic seizures. Such devices should detect seizures fast and with high accuracy, while respecting the strict energy budget on which they operate. Previous work has overlooked one or more of these requirements, resulting in solutions which are not suitable for continuous closed-loop stimulation. In this paper, we perform an in-depth design space exploration of a novel seizure-detection algorithm, which uses a complex Morlet wavelet filter and a static thresholding mechanism to detect absence seizures. We consider both the accuracy and speed of our detection algorithm, as well as various trade-offs with device autonomy when executed on a low-power processor. For example, we demonstrate that a minimal decrease in average detection rate of only 1.83% (from 92.72% to 90.89%) allows for a substantial increase in device autonomy (of 3.7x) while also facilitating faster detection (from 710 ms to 540 ms)

24. V. Prevelakis and M. Hamad, **"Controlling Change via Policy Contracts"**, *Proceedings of Internet of Things Software Update Workshop (IoTSU),* 2016.

    Relevance to SHARCS: This work discusses a policy-based admission and access control framework for software components. Specifically the paper introduces a flexible policy definition framework for determining whether updated software components should be admitted into an operational safety-critical and distributed platform (e.g. car, airplane, space-based platform). In addition the same mechanism can be used to ensure that the component continues to comply with its policy after admission. This framework will form the basis for the policy-based access control for libraries that TUBS is currently working on.

    Abstract: We introduce a flexible policy definition mechanism for determining whether updated software components should be admitted into an operational safety-critical and distributed platform (e.g. car, airplane, space-based platform). In addition the same mechanism can be used to ensure that the component continues to comply with its policy after admission.

2 Talks, seminars and presentations

As part of the broader dissemination effort for SHARCS, we presented various aspects of the SHARCS project at venues attracting the interest not only of the security community, but the wider research community as well.

1. Poster at HiPEAC conference, Prague, 18-19 January 2016

   The SHARCS project poster was presented in the HiPEAC conference poster session, which took place in Prague from 18th to 19th of January (see Appendix 2).

2. Sotiris Ioannidis and Vassilis Prevelakis organized the First International Workshop on Hardware Enhancements for Secure Embedded Systems (HESES), which was held as a meeting with the HiPEAC conference in Prague, Czech Republic, 18-19 January 2016. It focused on software support for embedded architectures, hardware–software synergies for secure systems, hardware security mechanisms and special security considerations in embedded systems security.

Figure 2.1: Prof. Vassilis Prevelakis (right) at HiPEAC - HESES 2016

In the context of the workshop, four presentations were given and a panel session was held:

- Presentation by Hannes Tschofenig, "ARM mbed OS: The End of the Flat Operating System (OS) Security Model", HESES Workshop, Prague, 18 January 2016
- Presentation by Sotiris Ioannidis (FORTH), "Secure Hardware-Software Architectures for Robust Computing Systems", HESES Workshop, Prague, 18 January 2016
- Presentation by Osman Sabri nsal, "Build it for Fault Tolerance, Get Security for Free", HESES Workshop, Prague, 18 January 2016
- Presentation by Francesco Regazzoni, "Bricks and Tools for Side Channel Resistant Hardware", HESES Workshop, Prague, 18 January 2016
- Panel session by Hannes Tschofenig, Ricardo Chaves, Osman Sabri nsal, Francesco Regazzoni, Vassilis Prevelakis, "Is security for embedded systems possible, or even desirable?", HESES Workshop, Prague, 18 January 2016

Figure 2.2: Giorgos Christou presenting the SHARCS poster at HiPEAC - HESES 2016

3. SHARCS hosted and supported the 9th European Workshop on Systems Security (EuroSec), which was planned as a full day event (with seven presentations and two key note talks) and was held in conjunction with the EuroSys conference, London, UK, April 2016.

4. FORTH presented SHARCS poster titled **"Leveraging DNS for timely SSL Certificate Revocation"** in womENcourage 2016 - 3rd ACM-W Europe Celebration of Women in Computing in Linz, Austria, in September 2016.

5. FORTH organized the 21st European Symposium On Research in Computer Security (ESORICS 2016), which was held in Heraklion, Greece, September 2016. SHARCS sponsored the event.

6. Christos Strydis gave a keynote presentation, *"Implant security: The new deep end"*, TRUEDEVICE Workshop, Dresden Germany, 2016

7. Cristiano Giuffrida gave a keynote presentation, *"Imagine a World Without Software Bugs (Hint: It Ain't that Pretty)"*, 2nd Cyber Security Workshop October 13, 2016, NWO, The Hague, The Netherlands

8. Sotiris Ioannidis gave a keynote presentation, *"Security applications of GPUs"*, International Symposium on Secure Virtual Infrastructures, Cloud and Trusted Computing (CTC), Rhodes, Greece, October 2016

9. Sotiris Ioannidis gave a keynote presentation, *"Security applications of GPUs"*, International Workshop on Information Security Theory and Practice (WISTP), Heraklion, Greece, October 2016 (in conjunction with ESORICS)

Figure 2.3: Keynote speaker Dr. Sotiris Ioannidis on "Security applications of GPUs" at WISTP 2016

10. There was a thematic session - Technological Challenges to IoT Security - at the 2016 HiPEAC Computing Systems Week in Dublin, in November, organized by Prof. Vassilis Prevelakis. This Thematic Session discussed the challenge of bringing improved security to Internet of Things and embedded systems in general.

    - Presentation by Professor Hamid Asgari, Thales Research and Technology (UK), *"A Programmable, Recursive, and Secure Network Architecture supporting diverse access and applications including IoT"*
    - Presentation by Dr. Stephen Farrell, Trinity College Dublin, *"Some Security and Privacy Challenges for Networks with Small Devices"*
    - Presentation by Sotiris Ioannidis (FORTH), *"Secure Hardware-Software Architectures for Robust Computing Systems"*

11. John Thomson (OnApp) was accepted at the Science Communication Event, held in Manchester, July 2016. During the event, aimed at Horizon 2020 coordinators, invited experts explained their views on science communication and particularly why they believe that communication is beneficial for researchers

Figure 2.4: Professor Hamid Asgari at HiPEAC CSW in Dublin

Figure 2.5: Dr. Stephen Farrell at HiPEAC CSW in Dublin

Figure 2.6: Sotiris Ioannidis at HiPEAC CSW in Dublin

Figure 2.7: Audience at HiPEAC CSW in Dublin

3 Academic Dissemination

3.1 Vrije Universiteit Amsterdam

Vrije Universiteit Amsterdam has currently four courses that are closely related to security and SHARCS-related topics:

1. **Secure Programming** (XB_40005, BSc, Year 3) allows students to familiarize with programming software that incorporates useful features for security purposes. During the course, students are exposed to APIs (OpenSSL) for cryptographic operations, such as symmetric/asymmetric encryption, cryptographic hashing, cryptographic protocols, digital certificates, encrypted sockets, and SSL/TLS.

2. **Computer and Network Security** (X_400127, MSc, Year 1) is a challenge-based course covering a wide spectrum of security issues. We explicitly focus on systems security to show students how attackers penetrate systems. Specifically, the course covers topics on (1) network security, (2) memory corruption and application security, (3) web security, (4) botnets, (5) cryptography.

3. **Secure Software** (XM_40019, MSc, Year 1) focuses on advanced exploitation techniques that can compromise software systems of different domains. Students need to complete 4 self-contained assignments in three different domains: (a) the Linux kernel, (b) a mobile device, (c) a web application and the browser. The fourth assignment targets understanding and implementing defenses that could potentially protect the systems in assignments (a)-(c).

4. **Binary and Malware Analysis** (X_405100, MSc, Year 2) deals with the hard problem of analyzing malware binaries, which, other than source code, does not provide human readable information on used data structures or the actual behavior of the code.

3.2 Chalmers Tekniska Högskola

Chalmers has a security specialization in its CSE graduate programs. It includes four courses related to security and SHARCS related topics:

1. **Cryptography** (TDA352) covers cryptographic primitives such as private-key and public-key ciphers, hash functions, MAC's and signatures and how to embed these in cryptographic protocols to achieve basic goals such as confidentiality, authentication and

non-repudiation, but also more elaborate services, such as key management, digital cash and electronic voting. Many examples of broken protocols are also discussed to enhance understanding of the engineering difficulties in building secure systems. Some of the above are related to the IMD SHARCS application.

2. **Computer Security** (EDA263) provides basic knowledge in the security area, i.e. how to protect systems against attacks. Attacks may change or delete resources (data, programs, hardware, etc), get unauthorized access to confidential information or make unauthorized use of the system's services. The course covers threats and vulnerabilities, as well as rules, methods and mechanisms for protection. Modeling and assessment of security and dependability as well as metrication methods are covered. A holistic security approach is presented and organizational, business-related, social, human, legal and ethical aspects are treated. Many parts of this course are also related to the SHARCS as it addresses system-level security issues.

3. **Language-based Security** (TDA602) dives into the principles of programming language-based techniques for computer security. Examining application-level attacks as races, buffer overruns, covert channels, and code injection as well as mastering the principles behind such language-based protection techniques as static analysis, program transformation, and reference monitoring. TDA602 is deals with higher-level application security topics which are only loosely related to SHARCS.

4. **Network security** (EDA491) covers security for networked applications, the weaknesses of communication protocols as well as countermeasures against attacks like firewalls and secure protocols like SSH and SSL. This is partly related to the network security requirements of the Cloud application.

3.3 Foundation for Research and Technology - Hellas (FORTH)

FORTH is a Research Center that has been contributing towards education via its student fellowship and practical training programs. FORTH engages a number of post-doctoral, doctoral, masters and undergraduate students in research and innovation projects. This is also the case in the SHARCS project. In this way, young scientists and early-stage researchers gain the skills necessary to succeed in the academic and business environment. Furthermore, permanent research staff from FORTH is supervising university students performing their undergraduate and graduate theses. Currently, one undergraduate student, seven postgraduates students and on phd student are performing their research in FORTH for SHARCS. Lastly, the university faculty employed at FORTH transfer knowledge to the university students through the regular courses taught as part of the curriculum.

3.4 Technische Universität Braunschweig

The Institute of Computer and Network Engineering (IDA) is internationally known as one of the pioneers in embedded system design automation, HW-SW co-design, and real-time analysis and optimization for networked systems. IDA has currently more than 60 staff members including many post doctoral researchers, and PhD students. The ever increasing importance of computer security in these areas means that there is a lot of interest within IDA and other collaborating groups of TUBS for graduate seminars with emphasis on security concepts and practices.

TU Braunschweig has integrated SHARCS concepts in the graduate course on Advanced Operating Systems in particular in the discussion of threat detection and containment as well in the presentation of how operating systems can leverage hardware features to improve performance. SHARCS was discussed in the graduate course on Security Topics in 2016. Moreover, TUBS has made SHARCS-based presentations in the graduate seminar program.

4 Other Dissemination Activities

During the period January 2016 – December 2016 the following additional dissemination activities were carried out:

1. Neurasmus issued one news article on the Erasmus Medical Center Monitor Magazine titled “Your heartbeat as password” (in Dutch: “Uw hartslag als wachtwoord”). http://www.erasmusmc.nl/5663/177341/211028/5634483/5634500/Monitor_2016-01.pdf (pages 16-17)

2. A press release was delivered by OnApp, on March 11th in France, Germany, Netherlands and Spain, detailing its involvement with the SHARCS project. http://onapp.com/2016/03/10/onapp-working-on-sharcs-ec/

3. The SHARCS project was referred in Anti-ROP blog on securityintelligence.com, as an EU-funded consortium targeted at providing end-to-end security across the hardware and software stack: https://securityintelligence.com/anti-rop-a-moving-target-defense

4. SHARCS was also mentioned in IBM Research Blog about the incorporation of AntiROP security: https://www.ibm.com/blogs/research/2016/04/its-always-harder-to-hack-attack-a-moving-target/

5. An overview of the SHARCS project was presented in the HiPEAC newsletter (HiPEAC Info 48, November 2016, p. 16-17) https://www.hipeac.net/assets/public/publications/newsletter/hipeacinfo48.pdf

6. The SHARCS leaflet was updated and will be handed out at HiPEAC, Stockholm-Sweden, January 2017 (see Appendix 1).

7. We produced a SHARCS project brochure and will hand it out at HiPEAC, Stockholm-Sweden, January 2017 (see Appendix 3).

8. We continued our on-line dissemination through the SHARCS web site and on the social media platforms Facebook and Twitter.

9. We established liaisons with related EU projects:

    - RAPID (H2020) - Joint journal publication Efficient Software Packet Processing on Heterogeneous and Asymmetric Hardware Architectures at IEEE Transactions On Networking

- 5G-Superfluidity (H2020). ONAPP collaborates on this project and is promoting the use of SHARCS techniques for ensuring security is pushed towards the end-points. ACTiCLOUD (H2020) - is a project where ONAPP is collaborating as the main hypervisor technology provider. The project focuses on pooling large, disaggregated memory resources and as such it is important to maintain security. To this effect ONAPP is looking at ensuring strong end-2-end security as advocated by SHARCS by providing security primitives at different layers.
- CyberSure (Marie Curie) - starts next year
- CCC (German) - Joint paper "Controlling Change via Policy Contracts" https://down.dsg.cs.tcd.ie/iotsu/subs/IoTSU_2016_paper_20.pdf at Internet of Things Software Update Workshop (IoTSU) 2016
- Dowser (Dutch), Parallax (Dutch), PASS (German)
- Rosetta (ERC) completed

5 SHARCS website and social media

5.1 Website Visitors & Trends

The sessions to the SHARCS website per month during the second year of the project can be seen in the Figure 5.1. We can see that a total of 4.408 sessions were recorded in this period. This means that we had an approximate of more than 13 visits per day.

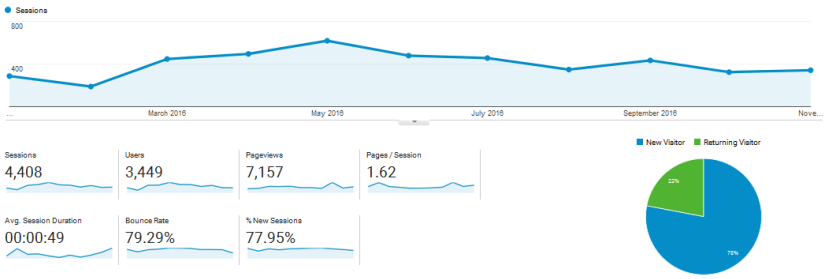


Figure 5.1: Visits to the SHARCS website.

The unique page views of the website appear in Figure 5.2 along with pages viewed by the visitors of our website which can be seen in Figure 5.3. Naturally enough, the front page of the website dominate the pageviews. It also seems that there is a considerable interest in our Partners and Publications section. Additionally, in Figure 5.4 you can see the geographic origin of the visitors.

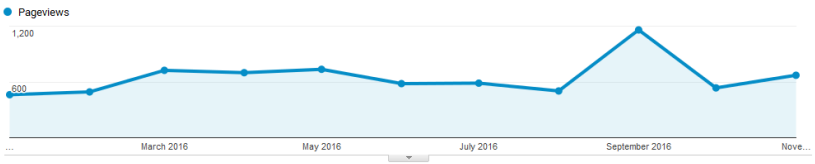


Figure 5.2: Unique page views of the website content.

Page	Pageviews	Unique Pageviews
	7,157 % of Total: 100.00% (7,157)	5,599 % of Total: 100.00% (5,599)
1. /	4,518 (63.13%)	3,670 (65.55%)
2. /publications/	911 (12.73%)	696 (12.43%)
3. /partners/	429 (5.99%)	368 (6.57%)
4. /workshops/heses-2016/	236 (3.30%)	175 (3.13%)
5. /events/	231 (3.23%)	136 (2.43%)

Figure 5.3: Unique page views of the website content per section.

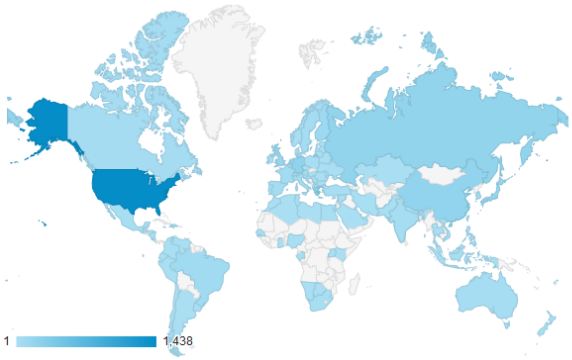


Figure 5.4: Geographic origin of the visitors of the project's website

5.2 Downloads per document category

To gain some insight on the preferences of our website visitors we categorized the documents we made available to the following categories:

- *Publications*: SHARCS sponsored articles and papers published by the consortium in peer reviewed conferences and journals.
- *Presentations*: Presentations made by the consortium in events related to the project.
- *Deliverables*: The deliverables produced by the project, as outlined in the description of work document.

Figure 5.5 shows how many times documents were downloaded from each category.

5.3 Twitter and Facebook

We proceeded to create a social circle around our project and its web page, by using both a Facebook page and a Twitter account linked to each other. Both were intensively used to

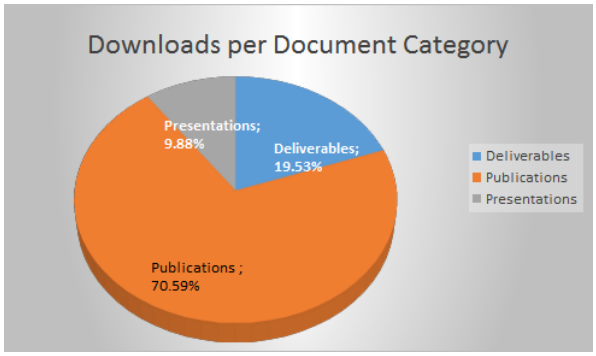


Figure 5.5: Downloads per document category.

disseminate event participations, openings of positions, published papers and news about SHARCS. The Facebook page has 89 followers (see Figure 5.6), whereas the Twitter feed is followed by 193 followers (102 more followers than the first year). In the second year of the project, we pushed 144 tweets through these channels (see Figure 5.7).

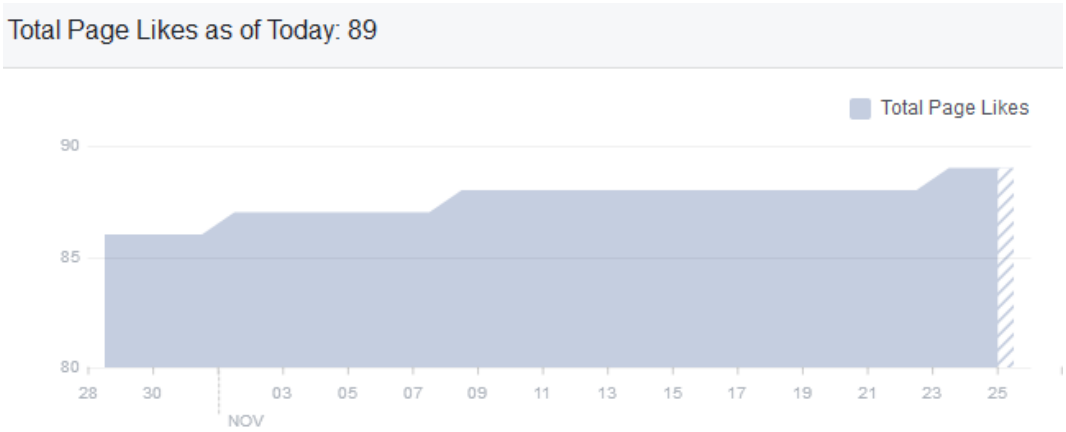


Figure 5.6: Likes of the Facebook page of the project.

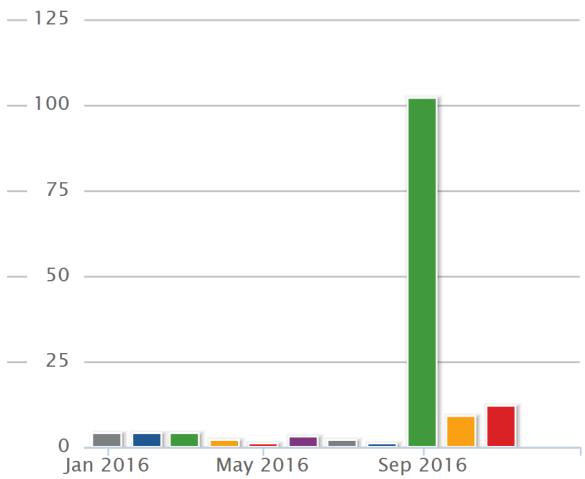


Figure 5.7: Tweets posted on the project's Twitter account.

6 Planned Dissemination

The following dissemination actions are planned for the SHARCS period January 2017 - December 2017:

- The workshop on Hardware Enhancements for Secure Embedded Systems (HESES) is planned as an event and will be held in conjunction with the HiPEAC conference, Stockholm, Sweden, January 2017
- At least five papers are planned for publication to top conferences or journals in the area in the period January 2017 - December 2017.

For completeness we list the following papers that have already been accepted for publication at the time this deliverable was finalized, which however do actually not belong to the time frame which is reported in this deliverable:

- B. Gras, K. Razavi, E. Bosman, C. Giuffrida, H. Bos, **"ASLR Cache: Practical Cache Attacks on the MMU from JavaScript"**, accepted for publication ain NDSS, 2017
- S. Rawat, V. Jain, A. Kumar, L. Cojocar, C. Giuffrida, H. Bos, **"Spelunker: Application-aware Evolutionary Fuzzing"**, accepted for publication in NDSS, 2017
- A. Milburn, C. Giuffrida, H. Bos, SafeInit, **'Comprehensive and Practical Mitigation of Uninitialized Read Vulnerabilities"**, accepted for publicationin NDSS, 2017
- A. Pawlowski, M. Contag, V. van der Veen, C. Ouwehand, T. Holz, H. Bos, E. Athanasopoulos, C. Giuffrida, **"MARX: Uncovering Class Hierarchies in C++ Programs"**, accepted for publication in NDSS, 2017
- X. Chen, H. Bos, C. Giuffrida, **"CodeArmor: Virtualizing the Code Space to Counter Disclosure Attacks"** accepted for publication EuroS&P, 2017

7 Year-2 Evaluation

In this chapter we compare the SHARCS dissemination activities for the second year against the goals described in first year's dissemination report.

Organization of two workshops: We organized our first workshop (First International Workshop on Hardware Enhancements for Secure Embedded Systems - HESES 16) which was held in January 2016, in Prague, Czech Republic. The second HESES workshop will be held on January 25th, 2017 in Stockholm, Sweden.

We also hosted and supported the 9th EuroSec workshop in London, UK, April 2016.

Participation in three networking events: Last year we took part in the TRUST conference and the EU sponsored CSP Forum. This year we took part in the TRUEDEVICE conference (March 18, 2016 - Dresden, Germany)

SHARCS also sponsored the ESORICS 2016 event in Heraklion, Greece, September 2016

Publication of at least twenty papers in conferences and journals: During this second year we had 24 publications.

Publication of at least three articles in the popular press: Neurasmus issued a news article on the Erasmus Medical Center Monitor Magazine. Furthermore, OnApp delivered a press release in March.

Creation of one brochure and one flyer that will be kept up to date throughout the project: We created one brochure and updated the flyer from the first year.

Establishment of liaisons and co operations with all related EU projects: We have established links with twelve other projects working in the same area as SHARCS.

Formation of one standardization proposal: Continued discussion with interested parties

Production of three promotional videos: Work in progress

Establishment and maintenance of a SHARCS presence on all relevant social media platforms: Our project has a sustained, continued presence in various social media platforms such as Facebook, Twitter, etc.

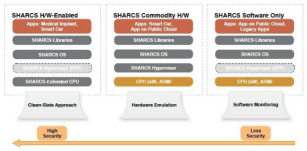
Provision of two related courses per year at each involved university partner: Except FORTH which is a research center and does not offer any courses, all other academic partners have at least two courses where SHARCS technologies were presented.

Formation of an External Advisor Board with at least six experts in the field: The External Advisor Board has been formed and a first meeting was held early 2016.

Appendix 1: The SHARCS 2016 updated leaflet

SHARCS Framework

SHARCS supports different framework operational models



Ideally, SHARCS pushes new functionality to the hardware level, and provides all necessary software-stack changes for producing and running hardened applications. However, modifying all levels is not always possible, therefore we provide two more relaxed SHARCS-supported models. First, one that incorporates zero hardware changes. For realizing this model all SHARCS features are communicated to a commodity processor (x86 or ARM) using a hypervisor. Second, one that incorporates zero SHARCS features implemented at the CPU, and there is no hypervisor available. For realizing this model, we link the application with SHARCS libraries and add kernel modules at the OS, which embed code for reliably and securely monitoring the application at run-time.

CANDIDATE TECHNOLOGIES

- Instruction Set Randomization
- Control Flow Integrity
- Information Flow Tracking
- Secure H/W Memory
- Fine-grained Memory Protection
- Dynamic Type Safety
- Model Checking of Software

SHARCS AT A GLANCE

PARTNERS

FORTH

OnApp Limited

Neurasmus BV

Chalmers Tekniska Hogskola

Vrije Univeristeit Amsterdam

Technische Universität Braunschweig

IBM Ltd

Elektrobit

PROJECT DETAILS

Start Date: 2015-01-01
Duration: 36 months
Project Cost: 3,105,762.50
Project Coordinator: FORTH

MORE INFORMATION

web: sharcs-project.eu
Twitter: @sharcs_project
Facebook: facebook.com/sharcsproject
Email: Sotiris Ioannidis, sotiris@ics.forth.gr

SHARCS

Secure
Hardware-Software Architectures
for Robust Computing Systems

OBJECTIVES

- Design, build, and demonstrate secure-by-design system architectures that achieve end-to-end security.
- Analyze and extend each hardware and software layer.
- Technologies developed directly utilizable by applications and services that require end-to-end security.

Supported
by the European Union
Horizon 2020 Programme
Under grant agreement
644571

SHARCS Applications

To demonstrate the efficacy and wide applicability of the SHARCS concepts, three real-world applications will be implemented as part of the project. These applications have been deliberately chosen from three different domains: medical devices, cloud applications, and smart cars.

Secure application execution on the (un-) trusted public cloud.

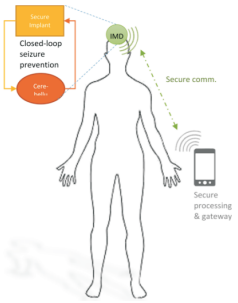
Our first application involves the trusted execution on an (un-)trusted public cloud. An end-user (EU) wants to make use of some software or services that are provided by an appliance-owner (AO). The software can be developed by the EU or more typically written by a third party application developer. The AO will buy datacenter resources either directly or via a cloud provider. If there is an issue with the service that is run then the issue may be at a single or many layers.

A DCO or CP that builds on a trusted platform will have some mechanism for exposing the security features. Procedures will be developed or re-used that allow for an AO to run a secured application on top of the platform. For the Public CP this requires a number of changes in the software platform used to expose resources. Trusted primitives will need to be exposed and then built on to allow more fully fledged services to be run. An AO may run some software that is SHARCS enabled as developed by an AD.

SHARCS Applications

Secure implantable neuromodulator for automatic seizure prevention

Epilepsy is a group of short - or long-term neurological conditions which are characterized by epileptic seizures. For the treatment of seizures, we are designing the prototype of a closed-loop, implantable neuromodulator that senses EEG signals, detects seizure onsets before they start and prevents them from manifesting through highly selective optogenetic stimulation of cerebellar regions.



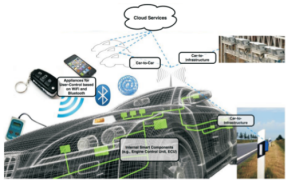
This application falls under the clean-slate operational model of SHARCS, enabling the integration of many novel hardware and software concepts. Work in SHARCS is focused on the implantable device which encompasses a novel security coprocessor and an assorted, lightweight security protocol in charge of wireless communication to external readers (e.g. Smartphones).

SHARCS Applications

This ensemble provides high security but also paves the way for zero-energy defenses in implants. SHARCS takes this concept further but adding extra layers of defense inside the implant SoC, such as low-power, hardware Instruction Set Randomization (ISR).

Secure Electronic Control Units (ECUs)

Vehicles are becoming ever more connected, and thus evolving into a mobile communication node in various directions. A smart car may communicate with other cars (Car-to-Car), with all kinds of infrastructure (Car-to-Infrastructure), with Cloud Services (e.g. real-time navigation or backup of settings) and with user appliances, such as smartphones, which can control it remotely. In this complicated setup all inter-connections and the components responsible for them (e.g. Electronic Control Units) must be secured.



Through SHARCS, the Electronic Control Units (ECUs), as well as the onboard and off-board communication, will be enhanced with secure hardware and software to prevent attackers from manipulating functionality or gaining unauthorized access to those ECUs. Such protection is essential for the safety and security of passengers on the road.

Appendix 2: The SHARCS poster at HiPEAC 2016

SHARCS: Secure Hardware – Software Architectures for Robust Computing Systems

Project Statement

SHARCS aims at designing, building and demonstrating secure-by-design system architectures that achieve end-to-end security for their users. SHARCS will achieve this by systematically analyzing and extending, as necessary, every hardware and software layer in a computing system.

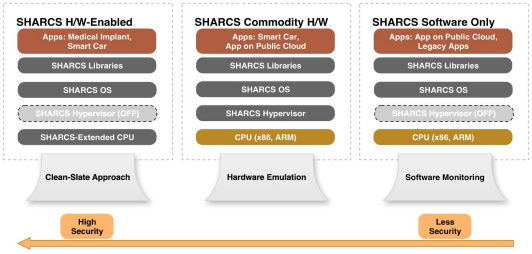
Key Issues and Objectives

- Extend existing H/W and S/W platforms towards developing secure-by-design enabling technologies
- Leverage H/W technology features present in today's processors and embedded devices to facilitate S/W- layer security
- Build methods and tools for providing maximum possible security-by-design guarantees for legacy systems
- Evaluate acceptance, effectiveness and platform independence of SHARCS technologies and processes
- Create high impact in the security and trustworthiness of ICT systems.
- Utilize new techniques including Instruction Set Randomization (ISR), Control Flow Integrity (CFI), Information Flow Tracking, Secure Hardware Memory Protection, Dynamic Type Safety, Model Checking of Software.

Project Details

Start Date: 2015-01-01 Duration: 36 months
Project Cost: € 3,105,762 Project Coordinator: FORTH

SHARCS Architecture



Operational models supported by the **SHARCS** framework include:

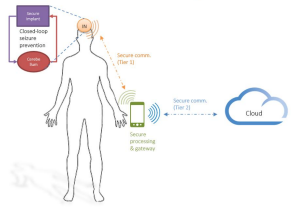
- **Clean-Slate Approach**: new hardware-based mechanisms with support from compilers, libraries and runtime environment.
- **Hardware Emulation** and a **SHARCS** hypervisor allow the new hardware-based security mechanisms to be made available on commodity architectures.
- **Software Monitoring**: The application is linked with **SHARCS** libraries and kernel modules which embed code for reliability and securely monitoring the application at runtime.

SHARCS Applications

Medical Devices

Secure implantable neuromodulator for automatic seizure prevention

For the treatment of seizures, we envision a closed-loop, fully implantable neuromodulator that senses EEG and single-neuron recordings, detects seizures before they start, and prevents seizure manifestation through highly selective optogenetic (or electrical) stimulation of brain regions.

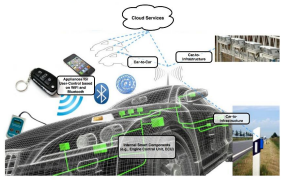


This application falls, as far as the implant device is concerned, under the clean-slate operational model of SHARCS, however many sub-components of the application follow the relaxed operational models. The implant will be enhanced with secure hardware and software so as to prevent malicious attackers from affecting implant functionality and/or from providing unauthorized access to the implant.

Smart Cars

Secure Electronic Control Units (ECUs)

Vehicles are becoming ever more connected, and thus evolving into a mobile communication node in various directions. A smart car may communicate with other cars (Car-to-Car), with all kinds of infrastructure (Car-to-Infrastructure), with Cloud Services (e.g. real-time navigation or backup of settings) and with user appliances, such as smartphones, which can control it remotely. In this complicated setup all inter-connections and the components responsible for them (e.g. Electronic Control Units) must be secured.



Through SHARCS, the Electronic Control Units (ECUs), as well as the onboard and off-board communication, will be enhanced with secure hardware and software to prevent attackers from manipulating functionality or gaining unauthorized access to those ECUs. Such protection is essential for the safety and security of passengers on the road.

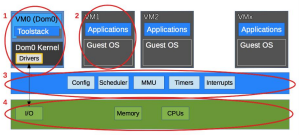
Clouds

Secure application execution on the (un-) trusted public cloud

Another application involves the trusted execution on an (un-) trusted public cloud.

Virtual Machine owners, use shared-tenancy hardware and hypervisor solutions provided by data centers to run their software applications in the Cloud.

The applications are run in isolation from each other but given the potential reward to attackers there have been many types of attacks exploited on virtualized platforms such as Amazon.



The current best of breed practice is to react to Xen Security Advisory (XSA) and other security patches as they are released. The SHARCS project will investigate secure-by-design approaches that look at securing;

A) Applications in an un-trusted Cloud Environment,

B) Securing the HV platform from bottom to top and presenting security primitives to VM owners.

SHARCS combines application and system focused approaches to ensure security by design.

Additional Information

web: sharcs-project.eu Facebook: facebook.com/sharcsproject Twitter: @sharcs_project Email: Sotiris Ioannidis, sotiris@ics.forth.gr

Supported by the European Union Horizon 2020 Programme under grant number 644571

Vrije Univeristeit Amsterdam

Technische Universität Braunschweig

Chalmers Tekniska Högskola

Appendix 3: The SHARCS brochure

Secure Hardware-Software Architectures for Robust Computing Systems

OBJECTIVES

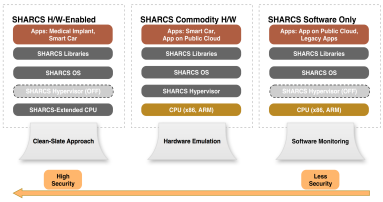
- Extend existing H/W and S/W platforms towards developing secure-by-design enabling technologies
- Leverage H/W technology features present in today's processors and embedded devices to facilitate S/W-layer security
- Build methods and tools for providing maximum possible security-by-design guarantees for legacy systems
- Evaluate acceptance, effectiveness and platform independence of SHARCS technologies and processes
- Create high impact in the security and trustworthiness of ICT systems

Supported by the European Union Horizon 2020 Programme under grant number 644571

(a)

SHARCS aims to build a framework for designing, building and demonstrating secure-by-design applications and services, which achieve end-to-end security for their users. This will be achieved this by systematically analyzing and extending the hardware and software layers in a computing system. This holistic approach is necessary, as no system can truly be secure unless every layer is secured. The effectiveness of the SHARCS framework will be measured by using it on a diverse set of security-critical, real-world applications that have been chosen from three completely different domains (medical, cloud, automotive) to demonstrate the platform-independence capabilities of SHARCS.

PROPOSED FRAMEWORK



SHARCS engages in research to fill security gaps, by offering a set of different models that are viable in different application domains. On the left, a full-featured SHARCS stack is introduced, which incorporates security changes in all layers. This model offers security guarantees for critical applications, control over critical infrastructure and follows a clean-slate approach. Obviously, this is not always feasible. There is no single *solution-for-everything*.

Therefore, SHARCS also proposes a more relaxed model utilizing standard hardware with zero changes. For realizing this model, all SHARCS features and modifications are communicated to a commodity processor (x86 or ARM) via a hypervisor. This relaxed model – which may be suitable for applications where enhanced hardware is not an option – offers the advantage that in some cases it may be applied as a software upgrade to existing hardware.

Last, a model is proposed that incorporates software-only changes, where neither the hardware nor hypervisor may be enhanced. For realizing this model, the application is linked with SHARCS libraries and kernel modules are added at the OS, which embed code for reliably and securely monitoring the application at run-time.

CANDIDATE TECHNOLOGIES

SHARCS utilizes new and existing hardware security features and propagates all needed changes to the software stack for supporting new as well as legacy applications. The hardware design and implementation of two fundamental security concepts – namely *Instruction Set Randomization* (ISR) and *Control-Flow Integrity* (CFI) – are introduced and a new policy is designed and implemented, which can be applied to legacy applications. SHARCS will take advantage of a few additional techniques.

Instruction Set Randomization ISR is a technique based on diversifying the language the execution environment understands, to prevent "foreign" code from executing. ISR proposes a runtime that can understand different randomized languages that an attacker cannot possibly know, so he is unable to execute his own code.

The runtime can be implemented directly in hardware or in software using virtualization. SHARCS implements ISR natively at the hardware level and provides all necessary software stack for generating ISR-compliant binaries, as well as existing code that can be reused through ISR-enabled SHARCS libraries.

(b)

Control-Flow Integrity Control-Flow Integrity (CFI) enforces a running binary to be contained by the developer's logic stemming from its actual source and not by a malicious one which can lead to software exploitation.

CFI thwarts control-hijacking attacks by ensuring that the control flow remains within the control-flow graph (CFG) intended by the programmer. Every instruction that is the target of a legitimate control-flow transfer is assigned a unique ID and checks are inserted before control-flow instructions to ensure that only valid targets from the list of known and allowable IDs are allowed.

SHARCS implements CFI support directly at the hardware level and modifies all software layers needed for exporting the functionality to running programs. It, also, offers a full compiler tool-chain for all the necessary modifications.

Information Flow Tracking Information-flow policies ensure that the propagation of data after it's been released, follows a specified policy. SHARCS adopts information-flow policies in applications where multiple entities exchange data across a multi-layer system, polices essential for guaranteeing that data is sufficiently constrained.

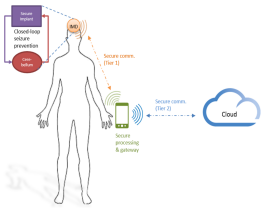
Secure H/W Memory Security can be easily bypassed because of simple implementation bugs. SHARCS adopts secure H/W memory for all sensitive data. Critical cryptographic keys are never stored in RAM, but in special H/W memory available only to the SHARCS platform or in existing H/W memory, such as CPU registers.

Fine-grained Memory Protection New additional features may be important for a system, however protection should be provided by constraining the new code from interfering with the rest of the system. Software Fault Isolation (SFI) is a strategy that isolates components of a process and prevents them from read/write actions outside an applied sandbox. SHARCS adopts H/W SFI based on segmentation where the underlying CPU allows that, or pure SFI where there is no support for segmentation.

Dynamic Type Safety Type systems are traditionally used to enforce and check correctness properties on whole classes of programs. These systems use either static, compile-time checking of security properties, or introduce run-time dynamic checks, or a combination of both. SHARCS explores ways to leverage the support for dynamic checks in existing hardware along with ways to remove expensive dynamic checks using type-based program analysis, for extended efficiency.

SHARCS APPLICATIONS

Secure, implantable neuromodulator for automatic seizure prevention



Epilepsy is a group of short- or long-term neurological conditions which are characterized by epileptic seizures. For the treatment of seizures, we envision a closed-loop, fully implantable neuromodulator that senses EEG and single-neuron recordings, detects seizure onsets before they start, and prevents seizure manifestation through highly selective optogenetic (or traditional) stimulation of cerebellar neurons.

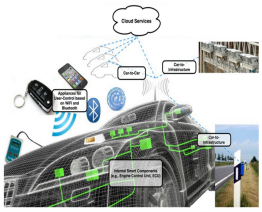
This application falls, as far as the implant device is concerned, to the clean-slate operational model of SHARCS, however many sub-components of the application follow the relaxed operational models. More precisely, both the implant and hand-held device will be enhanced with secure hardware and software (compiler) so as to prevent malicious attackers from affecting implant functionality and/or providing unauthorized access to implant.

(c)

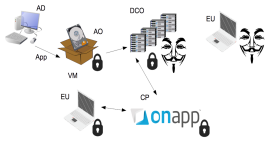
Secure Engine Control Units (ECUs)

SHARCS will also investigate secure applications running on a smart car interconnected with various communication components. A smart car may communicate with other cars (Car-to-Car), with the manufacturer's infrastructure (Car-to-Infrastructure), with Cloud Services (e.g., real-time navigation or backup of settings), and with user-appliances, such as smart- phones, which can control it remotely. In this complicated setup all interconnections and the components responsible for them must be secured. Furthermore, the smart car includes a series of digital smart components, such as the Engine Control Unit (ECU), which must be protected from malicious activities.

Through SHARCS onboard and off-board communication partners as well as mobile devices communicating with the vehicle will be enhanced with secure hardware and software to prevent attackers from manipulating functionality or gaining unauthorized access to those ECUs. Such protection will be essential for the future safety and security of passengers on the road.



Secure application execution on the (un-) trusted public cloud



Another application involves the trusted execution on an (un-)trusted public cloud. An end-user (EU) wants to make use of some software or services that are provided by an appliance-owner (AO). The software can be developed by the EU or more typically written by a third party application developer. The AO will buy datacenter resources either directly or via a cloud provider. If there is an issue with the service that is run then the issue may be at a single or many layers.

A DCO or CP that builds on a trusted platform will have some mechanism for exposing the security features. Procedures will be developed or re-used that allow for an AO to run a secured application on top of the platform. For the Public CP this requires a number of changes in the software platform used to expose resources. Trusted primitives will need to be exposed and then built on to allow more fully fledged services to be run. An AO may run some software that is SHARCS enabled as developed by an AD.

PROJECT DETAILS

Start Date: 2015-01-01
Duration: 36 months
Project Cost: 3,105,762.50
Project Coordinator: FORTH

MORE INFORMATION

web: sharcs-project.eu
Twitter: @sharcs_project
Facebook: facebook.com/sharcsproject
Email: Sotiris Ioannidis, sotiris@ics.forth.gr

(d)