Preference Proposals

- Each student will submit Two (2) "votes" for topic areas in the form of Two (2) "Preference Proposals"
- Deadlines:
  - Submit by EOD on Mon, September 24th
  - Discussion on Wed, September 26th.
  - No Class on Fri, September 28th
  - First Student Presentation: Wed, October 3rd
- Contents:

1. Select a Topic Area
2. Introduce a specific subject in the Topic Area
3. Describe an open question / problem on the subject
4. Describe a potential research project to address it
5. Cite 3+ papers on the subject (one of which may be the paper you are assigned to present).

Preference Proposals

Instructions:

- Written in LaTeX (template will be provided)
- Citations must be done in Bibtex
- Submit 2 PDFs on Compass2g
- Filename: <NetID>_<Topic Area #>_<Preference #>.pdf
- <Preference #>: 1= First Choice, 2 = Second Choice
- <Topic Area #>:

1	Foundations
2	Web Privacy
3	System Intrusions
4	Security Measurement
5	Mobile & Device Security
6	Human Factors

Preference Proposals

- Template
  - Title:

CS563: Advanced Computer Security
Preference Proposal
Topic Area Number: <Topic Area #>
Preference Number: <Preference #>
<Your name>
<Your NetID>

  - Introduction (Describe Subject)
  - Problem (Describe Open Question/Challenge)
  - Proposed Approach (Pitch potential project)
  - References

Example Preference Proposal #1

I. Introduction

Isolation between processes is at the core of many secure systems today. Clearly, applications which process sensi- tive information should not, in general, share memory with untrusted applications. Increasingly, however, system designers may wish to explicitly prevent any communication at all between two processes. For example, in a cloud computing environment, two independent virtual machines should not be able to communicate without explicit poli- cies allowing them to do so. One can imagine a scenario in government or industry in which a hypervisor runs multiple virtual machines (VMs) at varying security levels. As a general principle, we want to disallow communi- cation between virtual machines at different security levels in order to prevent information leakage. This is difficult to achieve, and has created an extensive back and forth between attackers, who seek to create covert channels to communicate information between processes, and defenders, who seek to design systems which guard against this sort of information leakage.

To the end of preventing unintended data disclosure from one process to another, recent years have seen the rise of systems which seek to provide secure enclaves, isolating the memory of one enclave from others and from the rest of the system. Intel SGX, ARM TrustZone, and MIT Sanctum are just a few of the most well known examples. These systems go to great lengths to offer isolation guarantees. They encrypt all data on-chip before sending it to memory, so even the operating system cannot access memory of applications running inside the enclave.

Although covert channels indeed present a challenge to system designers seeking to provide isolation guarantees, there are also concerns when only one party is malicious. These systems claim that external processes, including the operating system itself, cannot violate the confidentiality or integrity of software running inside an enclave. This property is fundamental to the trust users place in software running in these enclaves when they provide secrets to them. Unintentional leakage of confidential information is just as bad, if not worse, than intentional disclosure via a covert channel. Systems like [1], and many others designed on top of SGX, depend on the suppesed total isolation of enclave memory. If an attacker can access secret data within the enclave from the vantage point of another process outside the enclave, or from the operating system itself, then the security of these distributed systems is totally undermined. Unfortunately, there has been an extensive body of research recently showing that cache attacks enabling two-party covert channels and one-party snooping are often possible. Section two outlines this research and underscores the problems it presents. Section three details my proposed approach to addressing some of these problems and closing some of theses channels for information leakage.

[Murley, Fall 2017]

Example Preference Proposal #1

II. Problem

Previous research has shown that through memory thrashing or cache flushing, processes may establish a timing- based channel which does not depend on any shared memory or dorect communication such as sockets. In some cases, these channels may even work in the context of the supposedly isolated enclaves provided by recent advances like Intel SGX and MIT Sanctum. Specifically, the work of [2] demonstrates a technique by which an attacker can use caches to establish inter-process communication, even across cores. To do this, they use the fact that a process running on a core can actually evict a cache line from the L1 cache of another core. This gives them more control over the content of the Lower Level Cache (LLC), which they ultimately use to establish their covert channel.

Another recent paper on SGX side channel attacks [3] demonstrates side channel attacks an adversary could carry out to violate the confidentiality of an SGX enclave. In this case, the authors look at several different avenues of attack, including DRAM, caches, page tables, and the TLB. They demonstrate a technique callled ”sneaky page monitoring,” which they show can significantly reduce the number of page faults caused by these types of attacks, and thus curb the resulting performance degradation. This is just another example of the vulnerability of SGX in particular to side channel attacks. The specific problem of cache attacks is explored in more depth by [4]. Here, the authors actually demonstrate the viability of a confidentiality attack on a SGX enclave via cache observation without interrupting enclave execution. Using a prime+probe based attack and the built in performance monitoring capabilities, researchers are able to extract a full RSA private key from inside of an SGX enclave. These types of attacks break a key assumption made by many systems which claim to offer security - namely that data cannot leak out of the enclave through some side channel. If a process has a means to leak data beyond simply sharing memory, then true isolation is not in place and the value of remote attestation and memory isolation in these system is heavily diminished.

III. Proposed Approach

For this project, I propose to work towards mitigating cache-based side channel attacks on SGX. There are many possible paths to take in order to accomplish this, but I believe the most promising relates to address space random- ization within enclaves. The reason that external observers can learn information about the content of the enclaves by observing memory accesses is that they are able to learn over time which memory locations contain certain pieces of data. By randomizing memory adderesses before each run of the software, I believe we can make it much more difficult for an adversary to infer confidential information inside the enclave. As noted in [4], there has been some previous work to this end, but there is still progress needed. SGX Shield seeks to add ASLR to SGX for the purpose of raising the bar to attackers trying to exploit bugs in SGX software. However, this does not accomplish what is needed to deny cache side channel attacks, because it only randomizes instruction memory, and not data (where secrets would presumably be stored).

Runtime randomization of data memory seems like a challenging problem, as noted in [4]. Since data objects can be allocated at runtime, and since there may be large data objects which need to be split up, this will take some careful design work. However, if this hurdle is overcome and an effective data ASLR scheme is implemented, it could prevent this type of attack on SGX (and similar) enclaves. An attacker who does not know which data is stored at which address will not know which cache lines are being used for the target data. Therefore, they cannot use the same kind of prime+probe-type attack to observe memory access patterns and extract secrets.

[Murley, Fall 2017]

Example Preference Proposal #1

III. Proposed Approach (Continued)

It is possible this approach might prove too difficult. For reasons listed above, ALSR for data seems like a very difficult problem. If it starts to become clear in the course of research that this is an infeasible approach, I think that a good alternative could be exploring ways to write software so as to prevent leaking information via these side channels. For example, a lookup table based on a piece of secret data often emerges as a fundamental problem. I believe I could conduct some kind of taint analysis to find memory access that are dependent on some confidential value. From there, it might even be possible to rewrite the software in such a way that memory accesses are secret-independent. No doubt there are issues here as well. Any tool must begin with an understanding of which data is actually sensitive, and must track it throughout program execution. This is not trivial, and because it it is application-specific, it may be difficult to generalize. However, I think it could be a more accessible alternative if ASLR proves to be impractical.

The third and final option I will propose is to simply conduct a broad survey of all possible mitigations. This would be a fallback option and would probably not involve the practical implementation of these measures. I would research the state of the art and compare/contrast approaches to defending against cache-based side channel attacks on SGX. It seems that most of the papers on this subject are attack-based, i.e., they present a side channel attack and possibly offer a mitigation specific to that specific attack. I think it would be useful to take a more holistic look at the problem and possible solution. This would be my worst case fallback option if I was unable to accomplish either of the two approaches listed above.

References

[1] T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, "Ryoan: A distributed sandbox for untrusted computation on secret data," USENIX Security, 2016.

[2] C. Maurice, C. Neumann, O. Heen, and A. Francillon, "C5: Cross-cores cache covert channel," in Proceedings of the 12th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9148, ser. DIMVA 2015. New York, NY, USA: Springer-Verlag New York, Inc., 2015, pp. 46–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-20550-2 3

[3] W. Wang, G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, and C. A. Gunter, "Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX," CoRR, vol. abs/1705.07289, 2017. [Online]. Available: http://arxiv.org/abs/1705.07289

[4] F. Brasser, U. Mu¨ller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi, "Software grand exposure: SGX cache attacks are practical," CoRR, vol. abs/1702.07521, 2017. [Online]. Available: http://arxiv.org/abs/1702.07521

[Murley, Fall 2017]

Example Preference Proposal #2

I. Introduction

Previous work demonstrates that machine learning models trained on healthcare data can be vulnerable to mem- bership inference attacks. Membership inference attacks are attacks that allow an adversary to determine whether or not a particular data record was used to train a specific machine learning model. Shokri et al. 2016 shows that black box models such as those available via Amazon ML and Google Prediction API are vulnerable to membership inference attacks. This means that information about private data used to train various machine learning models is leaked by those models, even when only black box access to them is available.[1]

As previously mentioned, healthcare data has been proven vulnerable to membership inference attacks. Genomic data can be seen as an extension of healthcare data, in that the privacy guarantees and policies surrounding healthcare data should also apply for genomic data. Genomic data is particularly sensitive because of its personally identifying nature, permanence for an individual and among families, and sensitivity with respect to the healthcare information it contains. Genomic data has billions of attributes per person, which distinguishes it from traditional healthcare data. Because of the fundamental differences between traditional healthcare data and genomic data, it is not necessarily the case that the two types of data will be vulnerable to the same types of threats. Because of the similar risk of privacy leakage and because healthcare data is vulnerable to membership inference attacks, the question of whether or not genomic data is vulnerable to the same types of attacks should be posed.

Machine learning can be used for genomic data in various ways. Because of the size of genomic data, machine learning cannot often be used arbitrarily. It is often used on segments of genomic data rather than the entire genome to learn specific things about those segments on a population level. Often, domain specific knowledge about the genome and other traits to be predicted (for example diseases) is used to choose appropriate applications of machine learning to achieve good performance [2]. Because particular subsets of the genome are chosen for model training, we must determine how much information is leaked about a particular person if a membership inference attack succeeds and an adversary discovers that segments of the person's genome were present in the training set of the machine learning algorithm. Some segments of the genome are more personally identifying than others, so this information leakage should vary across tasks.

Machine learning tasks for genetic medicine can model a variety of different biological attributes including quan- tities of certain molecules in the cell, gene expression, splicing, and proteins binding to nucleic acids [3]. These modeling tasks take in genomic data as input and attempt to predict disease risk based on the biological attributes listed above. Large datasets help with these tasks because they allow researchers to learn more about risks for dis- eases across populations, but also have the risk of leaking information about a larger number of people. With modern computational resources, machine learning tasks on large scale genomic datasets are more frequent so analyzing the threat of privacy leakage is an important issue.

[Kaminsky, Fall 2017]

Example Preference Proposal #2

II. Problem

I propose a project to examine the amount of data leaked by black box machine learning algorithms about individuals whose genomic data is used in the training sets of the algorithms. More specifically, I want to explore if genomic data is vulnerable to membership inference attack, and to what degree. For example, if a portion of a person's genome is used in the training set of a black box machine learning algorithm, is it possible to infer membership of that genome segment in the training set? Further, is it then possible to identify the person to whom the portion of DNA belongs? The project will attempt to quantify the amount of data leaked about a particular individual based on a membership inference attack on his/her genome data in different scenarios.

III. Proposed Approach

I propose to follow the approach for membership inference attack introduced by Shokri et al. 2016. This approach depends on the fact that machine learning algorithms behave differently when used to predict examples from the training set than when used to predict new or unknown examples. The approach trains an attack model to distinguish data in the training set from data not in the training set based on the output of the machine learning model. To train the attack model, multiple "shadow" models are trained based on the black box algorithm. These shadow models are used because the training sets for them are known (they are generated to be similar to the real input data). Supervised training on the inputs and the corresponding outputs of the shadow models is then used for training the attack model to determine if the output of the shadow models is from an input that is a member of the training set or not [1]. For my project, I will follow this technique to develop attack models for a variety of machine learning tasks for learning information from genomic data. I will then evaluate if these tasks are vulnerable to membership inference attack using the attack models. If the genomic data in the training sets of the algorithms used for these tasks is vulnerable as a result of membership inferenece attack, I will attempt to determine the risks to the individuals to whom the data belongs.

References

[1] Reza Shokri et al. "Membership Inference Attacks Against Machine Learning Models". In: (2016).

[2] Maxwell W. Libbrecht and William Stafford Noble. "Machine Learning in Genetics and Genomics". In: (2015).

[3] Michael K. K. Leung et al. "Machine Learning in Genomic Medicine: A Review of Computational Problems and Data Sets". In: (2016).

[Kaminsky, Fall 2017]