Introduction to data confidentiality

damien.francois@uclouvain.be

Aim of the session

- ~~answer~~ discuss questions like
  - "What are best practice to work with sensitive and/or personally identifiable data?"
  - "What precautionary measures do I need to take to ensure data privacy?"
  - "How is confidentiality ensured on the clusters?"
- bring up awareness on challenges and tools

Example concerns

-  fMRI clinical data for machine learning research
- 🎙️ interview records for sociology research
- 🧪 materials research for an industrial partner

Disclaimers

- Sysadmins cannot answer the question "are my data sensitive?" or "what are the best practices?"
- Confidentiality requires security, and security is a shared duty
- Fireproof confidentiality would require major infrastructure adaptation

Are my data sensitive? What are best practices?

What makes data sensitive?

- 📚 Legislation (RGPD et al.)
- 📄 Agreements (NDAs)

Who can answer:

- 👩‍💼 Lawyers (Data Protection Officers)
- 👩‍🔬 Signees of the agreements (Principal Investigators)

There are multiple levels of confidentiality requirements:

- to *abstain*, and only work on 'local' hardware ; or
- to work only on *aggregations*, or intermediate results
- to *anonymize* the data
- to *pseudonymize* the data
- to *encrypt* the data, either
  - in transit (always the case)
  - at rest on disks (user duty)
  - at work in RAM (very difficult) ; or
- to *protect* the (clear) data from unauthorized access.

Data protection

Challenges

1. Clusters are designed primarily for performances and usability, but not confidentiality and security
    - Allowing users to build and run custom code
    - Maximizing uptime vs. frequent updates
    - Uniqueness of the hardware (e.g. GPUs) can be issue
    - Batch system does not allow interactive authentication

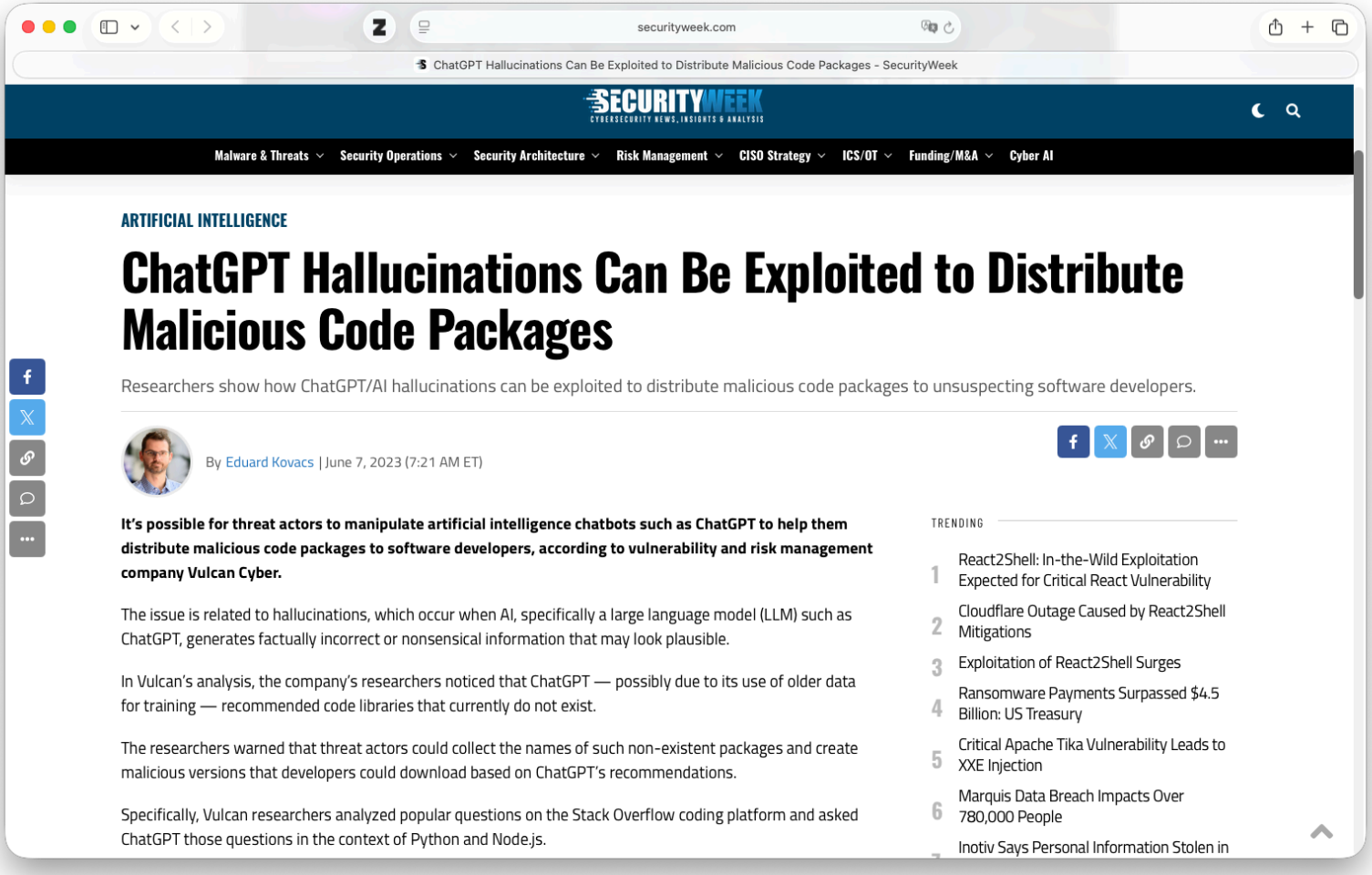
Challenges

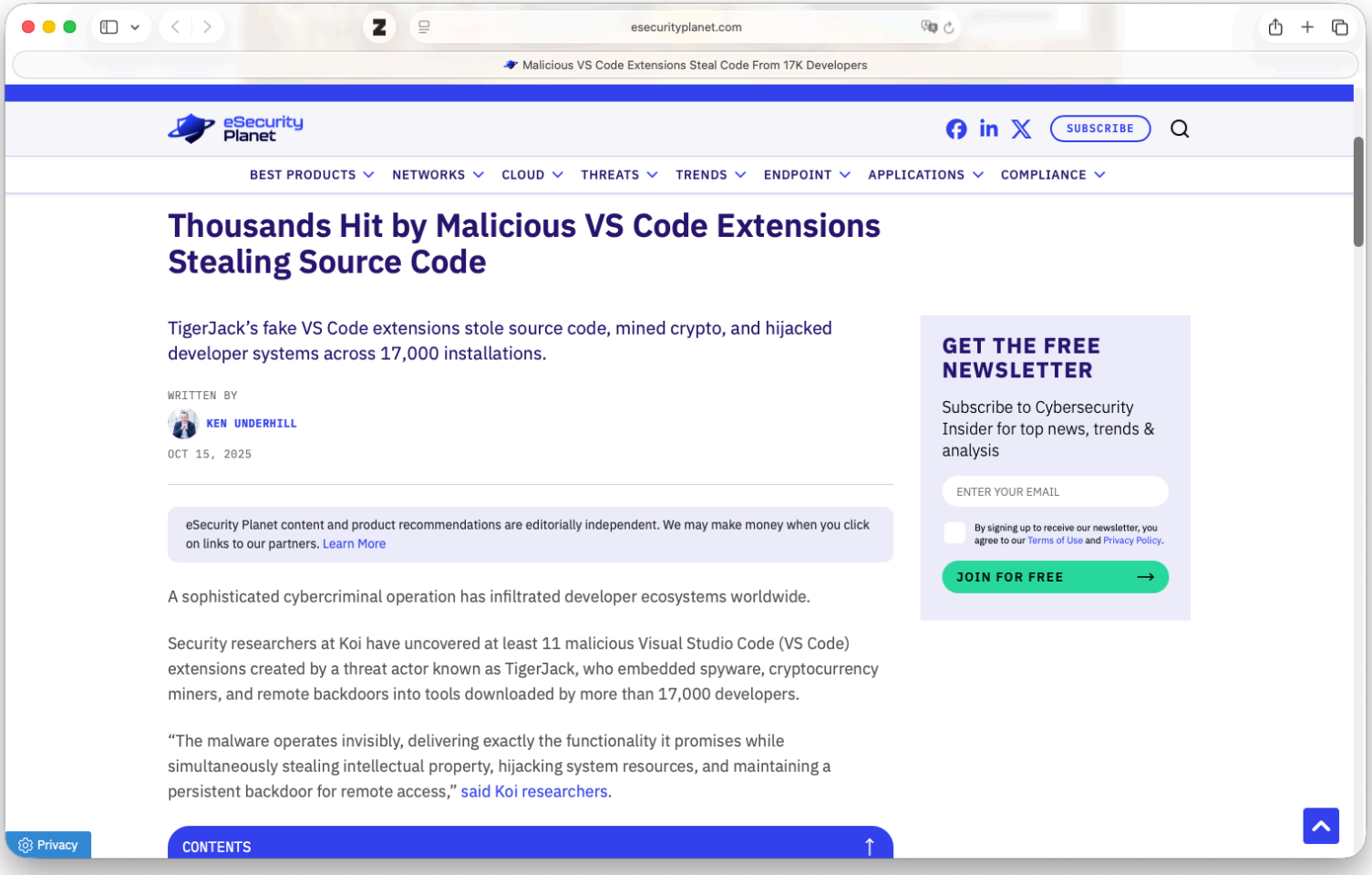
2. UNIX itself is not primarily designed for confidentiality
    - Process names, logins, default file permissions, port access are "public data"
    - Hardening methods harm performance and usability
    - User `root` has god-like powers

User responsibilities

Protect the infrastructure:

- Do not share credentials (SSH keys)
- Do not use "magic through firewalls" tools (i.e. third-party based) for file transfer or terminal sharing
- Beware of the tools you install (supply chain attacks)

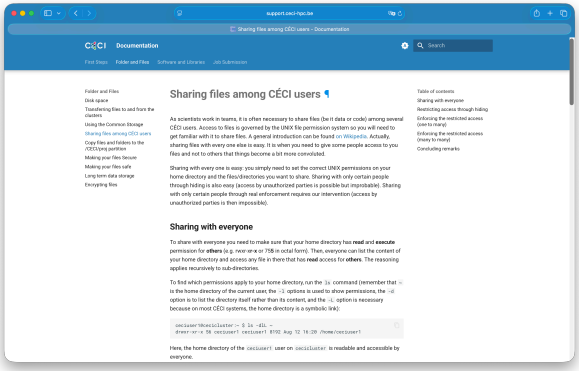

securityweek.com
ChatGPT Hallucinations Can Be Exploited to Distribute Malicious Code Packages - SecurityWeek
SECURITYWEEK
CYBERSECURITY NEWS, INSIGHTS & ANALYSIS
Malware & Threats
Security Operations
Security Architecture
Risk Management
CISO Strategy
ICS/OT
Funding/M&A
Cyber AI
ARTIFICIAL INTELLIGENCE
ChatGPT Hallucinations Can Be Exploited to Distribute Malicious Code Packages
Researchers show how ChatGPT/AI hallucinations can be exploited to distribute malicious code packages to unsuspecting software developers.
By Eduard Kovacs | June 7, 2023 (7:21 AM ET)
It's possible for threat actors to manipulate artificial intelligence chatbots such as ChatGPT to help them distribute malicious code packages to software developers, according to vulnerability and risk management company Vulcan Cyber.
The issue is related to hallucinations, which occur when AI, specifically a large language model (LLM) such as ChatGPT, generates factually incorrect or nonsensical information that may look plausible.
In Vulcan's analysis, the company's researchers noticed that ChatGPT — possibly due to its use of older data for training — recommended code libraries that currently do not exist.
The researchers warned that threat actors could collect the names of such non-existent packages and create malicious versions that developers could download based on ChatGPT's recommendations.
Specifically, Vulcan researchers analyzed popular questions on the Stack Overflow coding platform and asked ChatGPT those questions in the context of Python and Node.js.
TRENDING
1 React2Shell: In-the-Wild Exploitation Expected for Critical React Vulnerability
2 Cloudflare Outage Caused by React2Shell Mitigations
3 Exploitation of React2Shell Surges
4 Ransomware Payments Surpassed $4.5 Billion: US Treasury
5 Critical Apache Tika Vulnerability Leads to XXE Injection
6 Marquis Data Breach Impacts Over 780,000 People
Inotiv Says Personal Information Stolen in


esecurityplanet.com
Malicious VS Code Extensions Steal Code From 17K Developers
eSecurity Planet
SUBSCRIBE
BEST PRODUCTS
NETWORKS
CLOUD
THREATS
TRENDS
ENDPOINT
APPLICATIONS
COMPLIANCE
Thousands Hit by Malicious VS Code Extensions Stealing Source Code
TigerJack's fake VS Code extensions stole source code, mined crypto, and hijacked developer systems across 17,000 installations.
WRITTEN BY
KEN UNDERHILL
OCT 15, 2025
eSecurity Planet content and product recommendations are editorially independent. We may make money when you click on links to our partners. Learn More
A sophisticated cybercriminal operation has infiltrated developer ecosystems worldwide.
Security researchers at Koi have uncovered at least 11 malicious Visual Studio Code (VS Code) extensions created by a threat actor known as TigerJack, who embedded spyware, cryptocurrency miners, and remote backdoors into tools downloaded by more than 17,000 developers.
"The malware operates invisibly, delivering exactly the functionality it promises while simultaneously stealing intellectual property, hijacking system resources, and maintaining a persistent backdoor for remote access," said Koi researchers.
GET THE FREE NEWSLETTER
Subscribe to Cybersecurity Insider for top news, trends & analysis
ENTER YOUR EMAIL
By signing up to receive our newsletter, you agree to our Terms of Use and Privacy Policy.
JOIN FOR FREE
Privacy
CONTENTS

Tools and Methods

- File System Permissions (`777` is never a solution)

```
chmod -R o-rwx .
chmod -R g+rX .
umask 0077
```



https://support.ceci-hpc.be/doc/ManagingFiles/SharingFiles/

Tools and Methods

- Request exclusive access to nodes

```
sbatch --exclusive
```

https://slurm.schedmd.com/sbatch.html#OPT_exclusive

- Prevent untrusted software from "calling home"

```
unshare -n
```

https://man7.org/linux/man-pages/man1/unshare.1.html

Tools and Methods

- Protect network access to open ports

```
tensorboard --path_prefix
rserver --auth-pam-helper
vncpasswd
```

Refer to the documentation pages of the webservices you start on compute nodes.

Data encryption

Challenges

0. Encryption in transit: SSH
1. Encryption at rest: what about the storage of the secret password/key in a batch system?
2. Encryption at work: not entirely a solved problem

On the cluster, the `root` account can impersonate any user…

Encryption at rest

Key-based/password-based Encryption

- `age` (for file decryption/encryption)

https://github.com/FiloSottile/age

- `gocryptfs` (for full directory encryption)

https://support.ceci-hpc.be/doc/ManagingFiles/encryption/

- `munge` (for secret sharing with other users)

```
[dfr@lm4-f001 ~]$ munge -u bvr -s "secretpassword"
MUNGE:AwQFAACeZlJvKPlrZV8mazfLBQPS0iF25qa37yyBRau0VMc/GsJn [...]
[dfr@lm4-f001 ~]$ unmunge <<<"MUNGE:AwQFAACeZlJvKPlrZV8mazfLBQPS0iF25qa37yyBRau0VMc [...]
unmunge: Error: Unauthorized credential for client UID=3000003 GID=3000003
```

```
[bvr@lm4-f001 ~]$ unmunge <<<"MUNGE:AwQFAACeZlJvKPlrZV8mazfLBQPS0iF25qa37yyBRau0VM[...]
secretpassword
```

Storage of the secret

- Steganography

```
secret = ''.join(format(i, 'b') for i in bytearray("password", encoding ='utf-8'))
with open("instrumented_code.py", "w") as new:
        with open("code.py") as f:
                for i, line in enumerate(f, start=1):
                        line.strip()
                        if i < len(secret) and secret[i]:
                                line += " "
                        new.write(line)
```

- Obfuscation

```
# Pyarmor 9.1.8 (trial), 000000, non-profits, 2025-09-05T09:45:41.220705
from pyarmor_runtime_000000 import __pyarmor__
__pyarmor__(__name__, __file__, b'PY000000\x00\x03\r\x00\xf3\r\00\x00@\x0+\x8... ]')
```

- One time passwords online service (Vault)

```
$ curl https://secretstore.com/get/azl-sdfs-ere
secret
$ curl https://secretstore.com/get/azl-sdfs-ere
> 404 Not Found
```

Encryption at work : Homomorphic encryption

Compute in a transformed space e.g. RSA multiplication:

- Choose two large prime numbers, say `p` and `q`.
- Calculate the product of primes, `n = p * q`.
- Choose encryption exponent `e`
- Calculate decryption exponent `d`

Public Key = `(n, e)` ; Private Key = `(n, d)`.

- Encrypt first number `C1 = (m1^e) (mod n)`
- Encrypt second number `C2 = (m2^e) (mod n)`
- Multiplying the two encrypted texts, `C1` and `C2`, gives:
- `(m1^e) (mod n) × (m2^e) (mod n) = ( (m1^e) × (m2^e) ) (mod n) = ( (m1 × m2) ^e) (mod n)`
- Which is the encryption of `m1*m2`

Homomorphic encryption

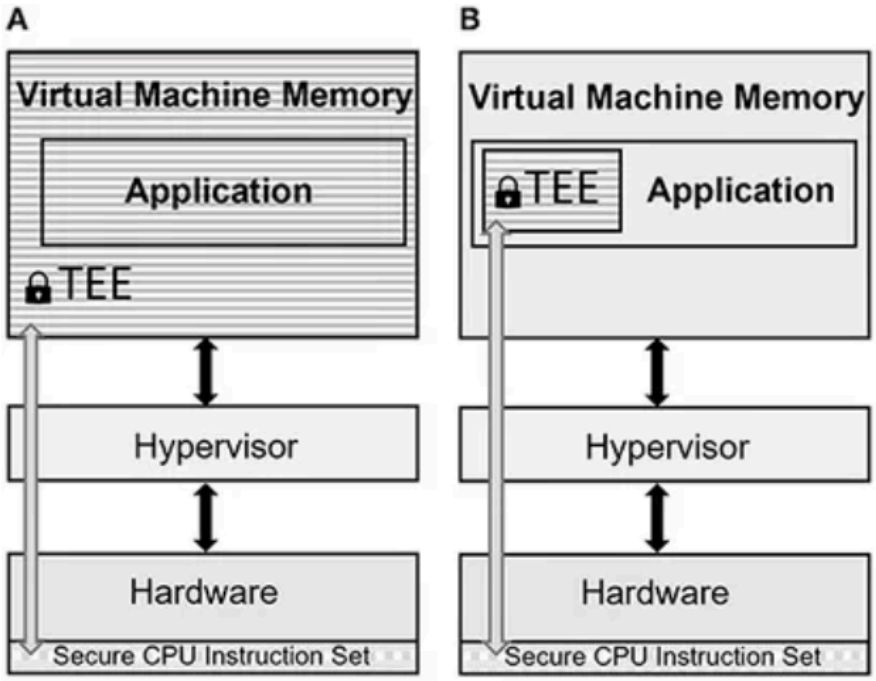
```
>>> import numpy as np
>>> from Pyfhel import Pyfhel
>>> HE = Pyfhel()
>>> HE.contextGen(scheme='bfv', n=2**14, t_bits=20)
'success: valid'
>>> HE.keyGen()
>>>
>>> integer1 = np.array([123], dtype=np.int64)
>>> integer2 = np.array([1], dtype=np.int64)
>>> ctxt1 = HE.encryptInt(integer1)
>>> ctxt2 = HE.encryptInt(integer2)
>>> print(ctxt1)
<Pyfhel Ciphertext at 0x14da90377560, scheme=bfv, size=2/2, noiseBudget=361>
>>> HE.decryptInt(ctxt1)
array([123,   0,   0, ...,   0,   0,   0])
>>>
>>> ctxtSum = ctxt1 + ctxt2
>>>
>>> resSum = HE.decryptInt(ctxtSum)
>>>
>>> print(resSum)
[124  0  0 ...  0  0  0]
```

https://pyfhel.readthedocs.io

https://github.com/jonaschn/awesome-he

Trusted Execution Environments

Figure 1

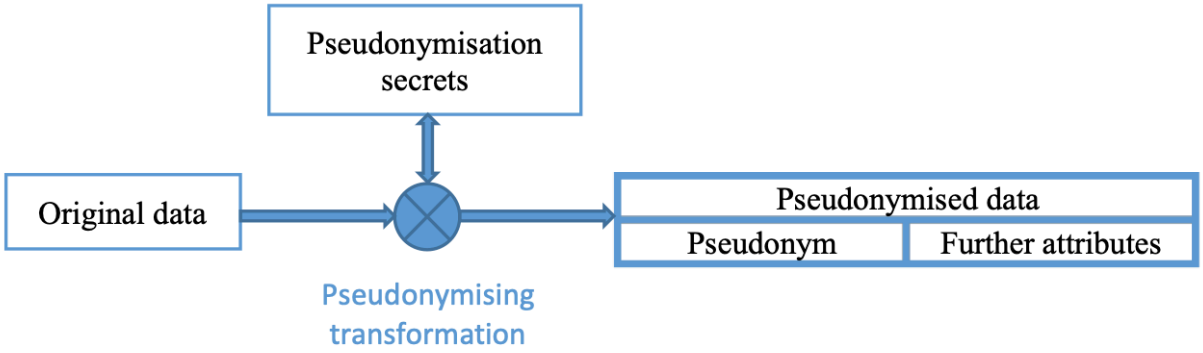


- Intel: Software Guard Extensions (SGX) ; Trust Domain Extensions (TDX)
- AMD: Secure Encrypted Virtualization (SEV)

Data pseudonimization

Pseudonimization ("reduce risks")

> both the concealment of the identifying information as well as the imposition of technical and organizational measures that prevent attribution to an identified or identifiable natural person without additional data



https://www.edpb.europa.eu/system/files/2025-01/edpb_guidelines_202501_pseudonymisation_en.pdf

Challenges

1. Storage of the pseudonimization secrets
2. Identification of PII in the data
3. Quasi-identifiers

> Variable values or combinations of variable values within a dataset that are not structural uniques but might be empirically unique and therefore in principle uniquely identify a population unit.

Tools and methods

Detection of PII:

- https://github.com/EdyVision/pii-codex
- https://github.com/DataFog/datafog-python

Do not use an online service for this task!

For quasi-identifiers:

- **Generalization**: e.g. use ranges instead of exact values
- **Data Swapping**: exchange values between records but keep marginal distributions
- **Differential Privacy**: add controlled mathematical noise with privacy guarantees

Data anonymization ("prevent re-identification")

Data Anonymisation

> process by which personal data is altered in such a way that a data subject can no longer be identified directly or indirectly, either by the data controller alone or in collaboration with any other party

Challenges

Same as for pseudonymisation, except must it must be bullet-proof.

Especially problematic with non-tabular data:

- speech data (voice identification, PII being spoken out, etc.)
- images of people (physical traits, geoguessing, etc.)
- free text (style recognition e.g. typos, PII being refered to in the text, etc.)
- ...

Tools and methods

- pitch alteration with e.g. with FFMPEG
- face detection in pictures with OpenCV
- authorship anonymization with adversarial stylometry
- ...

Data aggregation/summarization

Data aggregation/summarization

The process of only sharing aggregate data, which is not related to a single individual.

Challenges

1. Reduces the scope of possible analyses
2. Requires a distributed infrastructure
3. More difficult to integrate a cluster in the processing

Federated computation

Method allowing multiple data owners to compute a common statistic without sharing their data.

For instance: compute federated variance. Thanks to

$$s^2 = \frac{\sum_{i=1}^{k} \sum_{j=1}^{n_i} \left(x_{ij} - \overline{\overline{x}}\right)^2}{\sum_{i=1}^{k} n_i} = \frac{\sum_{i=1}^{k} n_i s_i^2}{\sum_{i=1}^{k} n_i} + \frac{\sum_{i=1}^{k} n_i \left(\overline{x}_i - \overline{\overline{x}}\right)^2}{\sum_{i=1}^{k} n_i}$$

multiple dataset owners can compute their subset variance and share it along with the subset size and mean. Then everyone can calculate the overall variance.

Federated computation

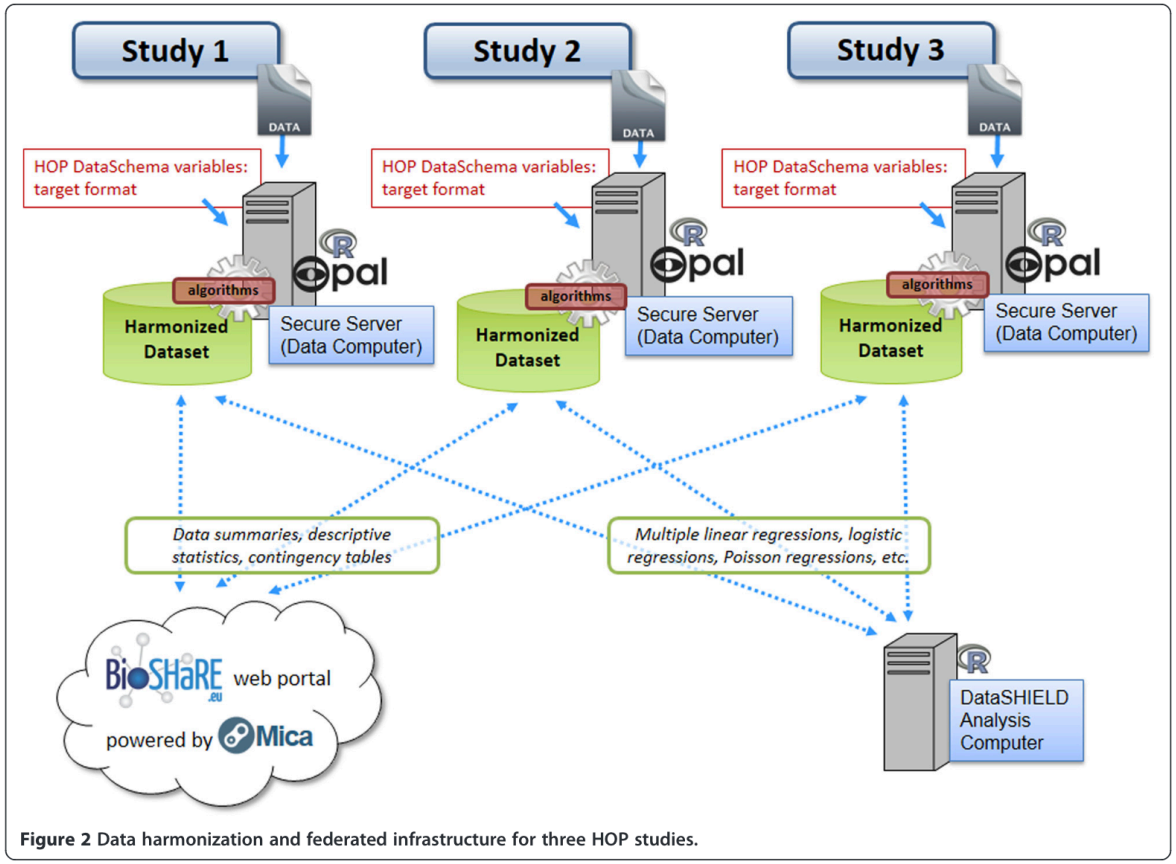


Figure 2 Data harmonization and federated infrastructure for three HOP studies.

Secure MultiParty Computation

Example: "Additive secret sharing": Alice, Bob, and Charlie want to calculate their average salary but not share them

- Everyone "splits" their salary into random sum

Alice: `40=44−11+7` - Bob `50=−6+32+24` - Charlie `60=20+0+40`

- Everyone shares the first two terms with the others
- Every one computes the sum of the half sum of the shared values with its third (secret) term
  - Alice can compute `(−6+32+20+0)/2 + 7 = 30`
  - Bob can compute `(44−11+20+0)/2 + 24 = 50.5`
  - Charlie can compute `(44−11−6+32)/2 + 40 = 69.5`
- Every one shares their results and sums them to get 150

Secure MultiParty Computation

```
from mpyc.runtime import mpc

async def main():
    secint = mpc.SecInt(16)

    await mpc.start()

    my_salary = int(input('Enter your salary: '))
    our_salaries = mpc.input(secint(my_salary))

    total_salary = sum(our_salaries)
    m = len(mpc.parties)

    print('Average  salary:', await mpc.output(total_salary) / m)
    await mpc.shutdown()

mpc.run(main())
```

Started 3 times with `python t.py –M3 –I$i`

Further reading

- Salomon, D., 2003. *Data Privacy and Security*. Springer New York, New York, NY.
- Venkataramanan, N., 2017. *Data privacy: principles and practice*. CRC Press, Boca Raton, FL.
- Jarmul, K., 2023. *Practical Data Privacy*, 1st ed. ed. O'Reilly Media, Incorporated, Sebastopol.