Exploring Synergies Between Privacy and Security Enhancing Technologies

David Klein

Institute for Application Security
Technische Universität Braunschweig

david.klein@tu-braunschweig.de

Preliminaries

- This is a sneak peak for a (so far) non public paper

⇒ David Klein, Benny Rolle, Thomas Barber, Manuel Karl, and Martin Johns. "General Data Protection Runtime: Enforcing Transparent GDPR Compliance for Existing Applications". In: *ACM CCS*. To appear. 2023

Preliminaries

- This talk covers privacy and some aspects of GDPR
- Some legal aspects

⇒ I'm not a lawyer!

Preliminaries

Setting:

- User of the software is not an adversary
  - ⇒ Wants to keep software secure
  - ⇒ Wants to comply with GDPR

Security Enhancing Technologies?

Example: Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email;

  db.saveEmail(email);

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

Example: Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email;
  
  db.saveEmail(email);

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

User controlled data enters application

Example: Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email; User controlled data enters application

  db.saveEmail(email); Save to storage

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

Example: Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email; User controlled data enters application

  db.saveEmail(email); Save to storage

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails(); Read from storage

  return res.render("emails", { emails: data });
});
```

Example: Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email; User controlled data enters application

  db.saveEmail(email); Save to storage

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails(); Read from storage

  return res.render("emails", { emails: data }); User controlled data is rendered
});
```

Example: Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email;

  db.saveEmail(email);

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

User controlled data enters application

Save to storage

Read from storage

User controlled data is rendered

Security Enhancing Technologies?

- **Dynamic Taint Tracking**

Security Enhancing Technologies?

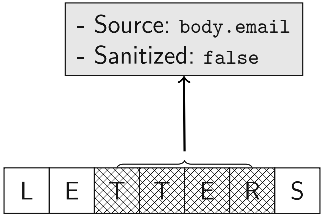
- **Dynamic Taint Tracking**
  - Attach labels to data

Security Enhancing Technologies?

- **Dynamic Taint Tracking**
  - Attach labels to data
  - Can prevent most “Injection Vulnerabilities” (A03 in OWASP Top 10 2021)

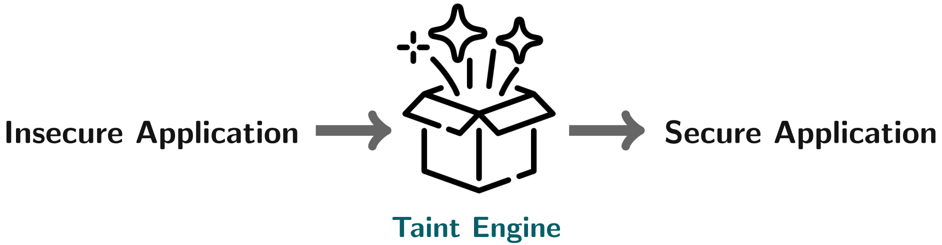
Security Enhancing Technologies?

- **Dynamic Taint Tracking**
  - Attach labels to data
  - Can prevent most “Injection Vulnerabilities” (A03 in OWASP Top 10 2021)



Security Enhancing Technologies?

- **Dynamic Taint Tracking**
  - Attach labels to data
  - Can prevent most “Injection Vulnerabilities” (A03 in OWASP Top 10 2021)



Preventing Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email;

  db.saveEmail(email);

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

Preventing Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email; Attach metadata

  db.saveEmail(email);

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

Preventing Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email; Attach metadata

  db.saveEmail(email); Persist metadata

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

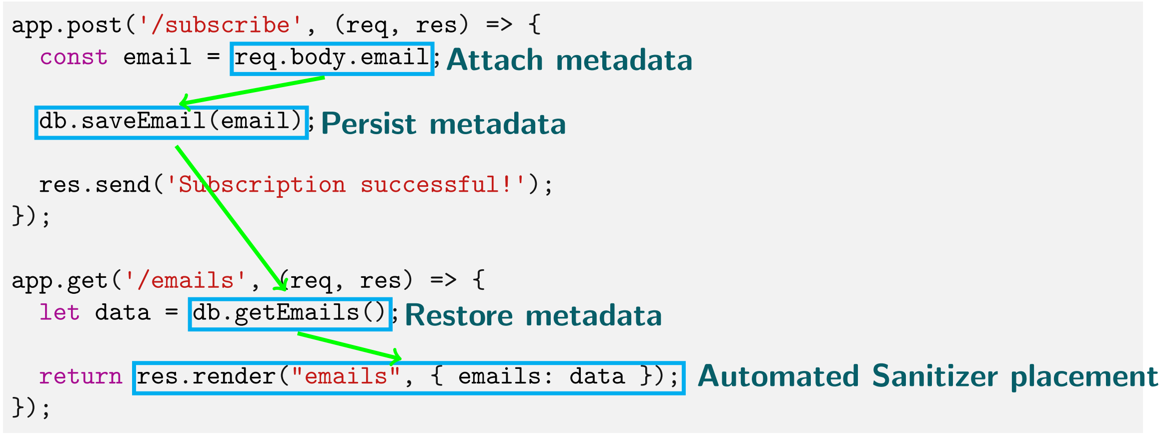
Preventing Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email; Attach metadata

  db.saveEmail(email); Persist metadata

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails(); Restore metadata

  return res.render("emails", { emails: data });
});
```

Preventing Stored Cross-Site Scripting

```
app.post('/subscribe', (req, res) => {
  const email = req.body.email; Attach metadata

  db.saveEmail(email); Persist metadata

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails(); Restore metadata

  return res.render("emails", { emails: data }); Automated Sanitizer placement
});
```

Preventing Stored Cross-Site Scripting



```
app.post('/subscribe', (req, res) => {
  const email = req.body.email;

  db.saveEmail(email);

  res.send('Subscription successful!');
});

app.get('/emails', (req, res) => {
  let data = db.getEmails();

  return res.render("emails", { emails: data });
});
```

From Security to Privacy

GDPR Violation

- In the context of this talk: **GDPR Violation = Violation of Purpose Binding**

GDPR Violation

- In the context of this talk: **GDPR Violation = Violation of Purpose Binding**
- Purpose Binding:

  Personal data shall be: collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes;. . .

  —GDPR Article 5(1)(b)

GDPR Violation

- In the context of this talk: **GDPR Violation = Violation of Purpose Binding**
- Purpose Binding:

  Personal data shall be: collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes;. . .

  —*GDPR Article 5(1)(b)*
- In my opinion among the most important articles

GDPR Violation

- In the context of this talk: **GDPR Violation = Violation of Purpose Binding**

- Purpose Binding:

  Personal data shall be: collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes;...

  —GDPR Article 5(1)(b)

- In my opinion among the most important articles
- If done right, determines what controller can do with PII

GDPR Violation

- In the context of this talk: **GDPR Violation = Violation of Purpose Binding**

- Purpose Binding:

  Personal data shall be: collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes;. . .

  —GDPR Article 5(1)(b)

- In my opinion among the most important articles
- If done right, determines what controller can do with PII
  - As seen in Simons talk, great success in the wild 😉

GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email;
  db.saveOrder( { ...
    email: email,
  });

  res.send('Purchase successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails();
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email;
  db.saveOrder( { ...
    email: email,
  });

  res.send('Purchase successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails();
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

$\boxed{\texttt{req.body.email;}}$ **PII entering application**

GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email; PII entering application
  db.saveOrder( { ...
    email: email, Storing PII
  });

  res.send('Purchase successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails();
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email; PII entering application
  db.saveOrder( { ...
    email: email, Storing PII
  });

  res.send('Purchase successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails(); Reading PII
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email; PII entering application
  db.saveOrder( { ...
    email: email, Storing PII
  });

  res.send('Purchase successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails(); Reading PII
  for (let email of data) {
    sendNewsletter(email); (Mis)using PII
  }
});
```

GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email; PII entering application
  db.saveOrder( { ...
    email: email, Storing PII
  });

  res.send('Purchase successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails(); Reading PII
  for (let email of data) {
    sendNewsletter(email); (Mis)using PII
  }
});
```

Those look fairly similar, right?

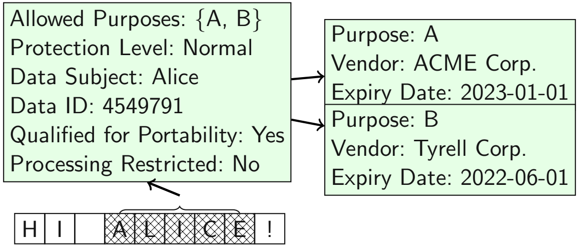
STORED XSS DATAFLOW
GDPR VIOLATION DATAFLOW
Corporate needs you to find the differences between this picture and this picture.
They're the same picture.
imgflip.com

Tainting for GDPR Compliance

- Important difference:
  - Deciding whether data flow is “unwanted” is much more involved for privacy
  - Requires more complex metadata

Tainting for GDPR Compliance

- Important difference:
  - Deciding whether data flow is “unwanted” is much more involved for privacy
  - Requires more complex metadata



GDPR Taint Metadata

Preventing GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email;
  db.saveOrder( { ...
    email: email,
  });

  res.send('Subscription successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails();
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

Preventing GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email;
  db.saveOrder( { ...
    email: email,
  });

  res.send('Subscription successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails();
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

Attach GDPR metadata

Preventing GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email;
  db.saveOrder( { ...
    email: email,
  });

  res.send('Subscription successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails();
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

$\boxed{\texttt{req.body.email;}}$ **Attach GDPR metadata**

$\boxed{\texttt{email: email}}$ **Persist metadata**

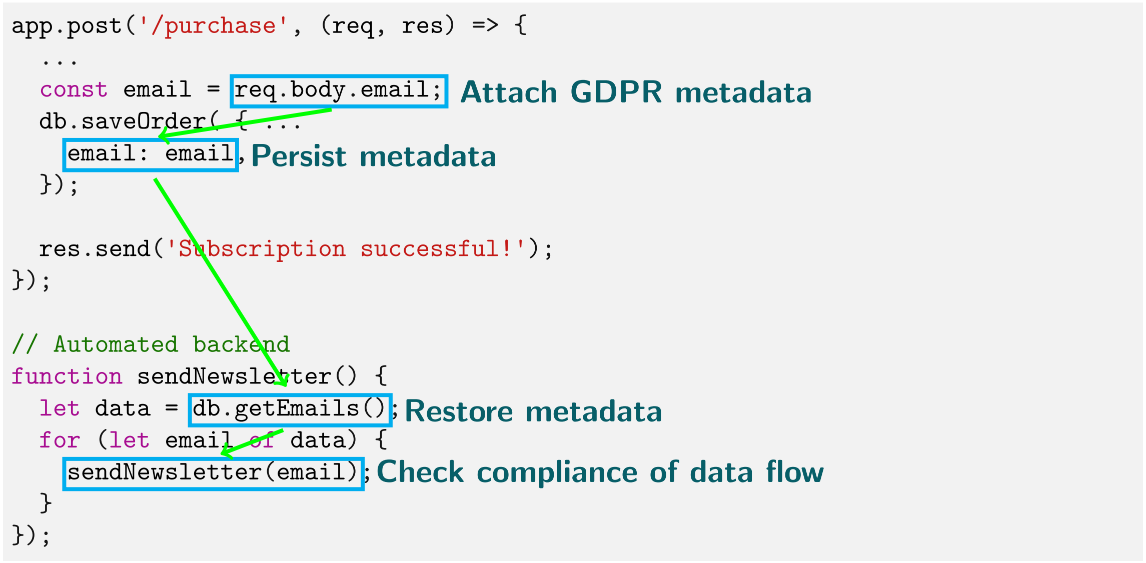
Preventing GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email; Attach GDPR metadata
  db.saveOrder( { ...
    email: email, Persist metadata
  });

  res.send('Subscription successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails(); Restore metadata
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

Preventing GDPR Violation

```
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email; Attach GDPR metadata
  db.saveOrder( { ...
    email: email, Persist metadata
  });

  res.send('Subscription successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails(); Restore metadata
  for (let email of data) {
    sendNewsletter(email); Check compliance of data flow
  }
});
```

Preventing GDPR Violation



```javascript
app.post('/purchase', (req, res) => {
  ...
  const email = req.body.email;
  db.saveOrder( { ...
    email: email,
  });

  res.send('Subscription successful!');
});

// Automated backend
function sendNewsletter() {
  let data = db.getEmails();
  for (let email of data) {
    sendNewsletter(email);
  }
});
```

Restoring metadata?

- Data flows going through storage historically problematic

Restoring metadata?

- Data flows going through storage historically problematic
  - ⇒ Most taint engines lose metadata

Restoring metadata?

- Data flows going through storage historically problematic
  - ⇒ Most taint engines lose metadata
- Prevents detection and prevention of complex Vulnerabilities

Restoring metadata?

- Data flows going through storage historically problematic
  - ⇒ Most taint engines lose metadata
- Prevents detection and prevention of complex Vulnerabilities

⇒ Persisting taints is essential

Restoring metadata?

- Data flows going through storage historically problematic
  - ⇒ Most taint engines lose metadata
- Prevents detection and prevention of complex Vulnerabilities

⇒ Persisting taints is essential

Solution:

- Rewrite SQL queries on the fly to persist metadata alongside data

Restoring metadata?

- Data flows going through storage historically problematic
  - ⇒ Common taint engines lose metadata
- Prevents detection and prevention of complex Vulnerabilities

⇒ Persisting taints is essential

Solution:

- Rewrite SQL queries on the fly to persist metadata alongside data
- Before:

```
UPDATE a = ? in tbl WHERE id = ?;
```

- After:

```
UPDATE a = ?, a_t = ? in tbl WHERE id = ?;
```

Why don't people use the magic tainting box?

Benchmark	Overhead
avrora	6.8%
batik	11.2%
biojava	104.4%
graphchi	-2.3%
luindex	7.2%
sunflow	-1.2%
zxing	5.6%
fop	33.8%
h2	111.2%
jme	1.1%
Average	27.8%

Why don't people use the magic tainting box?

Benchmark	Overhead
avrora	6.8%
batik	11.2%
biojava	104.4%
graphchi	**-2.3%**
luindex	7.2%
sunflow	**-1.2%**
zxing	5.6%
fop	33.8%
h2	111.2%
jme	1.1%
Average	27.8%

Why don't people use the magic tainting box?

Benchmark	Overhead
avrora	6.8%
batik	11.2%
biojava	**104.4%**
graphchi	-2.3%
luindex	7.2%
sunflow	-1.2%
zxing	5.6%
fop	33.8%
h2	**111.2%**
jme	1.1%
Average	27.8%

Why don't people use the magic tainting box?

Benchmark	Overhead
avrora	6.8%
batik	11.2%
biojava	104.4%
graphchi	-2.3%
luindex	7.2%
sunflow	-1.2%
zxing	5.6%
fop	33.8%
h2	111.2%
jme	1.1%
Average	**27.8%**

Summary

- We built a prototype realizing concepts presented today
  - for “arbitrary” Java applications
  - Collaboration with SAP Security Research
  - called **Fontus**

Summary

- We built a prototype realizing concepts presented today
  - for “arbitrary” Java applications
  - Collaboration with SAP Security Research
  - called **Fontus**
- Concept is generic and not reliant on our prototype

Summary

- We built a prototype realizing concepts presented today
  - for "arbitrary" Java applications
  - Collaboration with SAP Security Research
  - called **Fontus**
- Concept is generic and not reliant on our prototype
- Attaching privacy metadata to data really powerful
  - Allows to automate e.g., Subject Access Requests

Summary

- We built a prototype realizing concepts presented today
  - for "arbitrary" Java applications
  - Collaboration with SAP Security Research
  - called **Fontus**
- Concept is generic and not reliant on our prototype
- Attaching privacy metadata to data really powerful
  - Allows to automate e.g., Subject Access Requests

⇒ **Security and Privacy despite Design**

Summary

- We built a prototype realizing concepts presented today
  - for "arbitrary" Java applications
  - Collaboration with SAP Security Research
  - called **Fontus**
- Concept is generic and not reliant on our prototype
- Attaching privacy metadata to data really powerful
  - Allows to automate e.g., Subject Access Requests

⇒ **Security and Privacy despite Design**

- What's the maximum overhead for people to apply tainting in production?

Thank you for your attention!

TESTABLE

CASA
CYBER SECURITY IN THE AGE OF LARGE-SCALE ADVERSARIES

Contact

✉ david.klein@tu-braunschweig.de
in david-klein-b2aa80254
twitter.com/ncd_leen