A Study of Interoperability in Electronic Health Record Software

Prashant Anantharaman*
Narf Industries
San Diego, California, USA

Rebecca ".bx" Shapiro
Narf Industries
San Diego, California, USA

Vishnupriya Varadharaju
Narf Industries
San Diego, California, USA

Michael E. Locasto
Narf Industries
San Diego, California, USA

Abstract

The healthcare industry has been the victim of numerous high-profile cyberattacks in recent years. Entire hospital networks have been compromised due to ransomware, forcing hospitals to temporarily revert to an entirely paper-based system while backups are restored or, worse, the ransom is paid. With the push for more interoperability between the Electronic Health Record Systems (EHRs) used by hospitals, standards such as Fast Health Interoperability Resources (FHIR) have come up to ensure standardized data exchange between these providers.

This paper studies the support for such interoperability protocols in popular open-source EHRs. We built a first-of-its-kind system, the FHIR Garden, that provides a containerized environment to compare several FHIR implementations by importing and exporting the same patient data across all the implementations. Unlike other interoperability-focused tools, our system does not find non-compliance or implementation-specific issues. Instead, we focus on finding mismatches between how implementations process and export a patient's records, which adversaries can then leverage to affect patient health.

As a result of this work, we identified 59 parser differentials in JSON implementations in popular open-source FHIR servers. We also identified vulnerabilities in OpenEMR related to their Consolidated Clinical Document Architecture format that were promptly disclosed and fixed.

CCS Concepts

• **Security and privacy → Web protocol security**; **Software reverse engineering**.

Keywords

Electronic Health Records, Health Information Exchanges, Fast Health Interoperability Resources (FHIR), Language-Theoretic Security

*Email: prashant.anantharaman@narfindustries.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
HealthSec '24, October 14–18, 2024, Salt Lake City, UT, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1238-8/24/10
https://doi.org/10.1145/3689942.3694743

ACM Reference Format:
Prashant Anantharaman, Rebecca ".bx" Shapiro, Vishnupriya Varadharaju, and Michael E. Locasto. 2024. A Study of Interoperability in Electronic Health Record Software. In *Proceedings of the 2024 Workshop on Cybersecurity in Healthcare (HealthSec '24), October 14–18, 2024, Salt Lake City, UT, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3689942.3694743

1 Introduction

Recent cyberattacks against clinical data processing systems, such as United Healthcare, have been estimated to have resulted in losses of almost $1 billion in a single incident [4], compared to the average cost of US-based cyber incidents estimated at $9.4 million [12]. These attacks pose a greater threat than normal to patients' privacy, given the nature of the data stored by hospitals, such as demographic information, medical histories, mental health conditions, insurance coverage information, and any other notes taken by a medical health professional. Given these risks, utmost attention is paid to protecting these critical systems from adversaries.

On the other hand, there are numerous situations where a patient may want their health records sent to another provider. (1) A patient may be switching primary care providers. (2) A patient needs to visit a specialist who needs to see the prior records. (3) A patient is receiving emergency care, and the providers need to know about prior conditions and allergies. These constitute scenarios where a hospital must facilitate the transfer of these health records to another provider to aid the patient. Health Information Exchanges (HIE) facilitate these transactions in a geographic region [13].

Before the 21st Century Cures Act passed, Medical Software was defined as a "Medical Device" regulated by the Food and Drug Administration (FDA) under Section 520 of the Federal Food, Drug, and Cosmetic Act. This categorization made it hard for electronic health record software to be updated with sufficient certification from the FDA. Subsequently, the passage of the Health Information Technology for Economic and Clinical Health Act in 2009 incentivized healthcare providers to adopt Electronic Health Record (EHR) software to record patient data to improve patient care and interoperability.[1] The US CDC reports that as of 2021, 88.2% of physicians use any EHR, whereas 77.8% of them use a certified EHR.[2]

An EHR is complex and includes over 5000 variables, with more being added every day [5]. A study of patient experiences with EHRs and interoperability in the National Health Service (NHS) in the United Kingdom [15] found that a lack of adequate interoperability

[1] https://www.govinfo.gov/content/pkg/PLAW-111publ5/pdf/PLAW-111publ5.pdf
[2] https://www.cdc.gov/nchs/fastats/electronic-medical-records.htm

led to inaccurate medical records and forced patients to identify workarounds to provide the needed clinical data rather than using an interoperability service. Numerous studies have also examined how introducing EHRs and interoperability has led to physician burnout [2, 9, 11, 23].

Interoperability Standards. Two standards organizations primarily specify healthcare interoperability requirements and rules. First, openEHR defines syntactic and semantic archetypes for clinical data [10]. openEHR is widely used by the National Health Services of the United Kingdom and in Australia by the National e-Health Transition Authority of Australia. The Health Level 7 (HL7) has provided several standards since its founding in 1989. Their core standards for interoperability started from version 2, version 3, Clinical Document Architecture (CDA), to the now popular Fast Health Interoperability Resources (FHIR; pronounced fire). These HL7 standards are text-based, where CDAs use XML, whereas FHIR supports XML, JSON, and Turtle formats.

Table 1 shows the number of publicly accessible FHIR servers by the vendor as searched on the Internet search engine Shodan [18]. Cloud-based providers, such as Microsoft, Google, Epic, and Cerner, may be sufficiently masking their FHIR instances or placing them inside virtual private networks (VPNs) to prevent them from being discovered using tools like Shodan. HAPI FHIR is the most popular open-source FHIR implementation and powers several commercial FHIR offerings, such as Smile CDR and OpenMRS.

FHIR Service	Instances on Shodan
Google FHIR	3
HAPI FHIR	36
Smile CDR	23
OpenEMR	1057
Microsoft FHIR	3
Health Intersections	3

Table 1: FHIR Server Instances publicly accessible

This paper seeks to understand the extent to which interoperability standards have been adopted in open-source software and how they withstand malicious data. Differences between implementations of the same data format, known as *parser differentials*, can have dire consequences for real-world systems. For example, numerous HTTP smuggling attacks and Apple property list vulnerabilities have been demonstrated in the past that focus on exploiting differences between parsers [14, 19]. In healthcare settings, differentials between how a FHIR record is interpreted by two implementations can lead to errors in patient records.

This study focuses on identifying misinterpretations between the JSON and XML implementations of various FHIR servers to understand how they adhere to the rules in the FHIR specification and whether violations of differences can be exploited. First, we found that adoption of FHIR interoperability standards in open-source EHRs are minimal, and they may often be error-prone. Next, we built the FHIR Garden, a collection of containers to compare open-source FHIR servers, and a toolkit to provide the same input to the servers and compare their output.

Contributions.

- We present a first-of-its-kind testbench to compare various FHIR implementations by sending the same request to all servers and comparing their output. This test bench is important for studying the interoperability of these EHR and FHIR software solutions.
- Our toolkit also supports chaining multiple FHIR servers using import/export operations to study how vulnerabilities can be leveraged.
- We present numerous findings demonstrating how specifications have been misinterpreted by prominent EHR software and how this affects interoperability.

Responsible Disclosure. Our findings were reported to the OpenEMR community through their GitHub repository, discussion board, and Telegram group.

Code Availability. Our code is available at https://github.com/narfindustries/digiheals-public under the GNU Public License Version 3.

2 Related Work

Fast Health Interoperability Resources (FHIR). The FHIR standard, introduced in 2011, is currently in its fifth release and provides rules for importing and exporting electronic health records. FHIR is implemented on top of HTTPS, where REST APIs are provided for participants to query. Some examples of resources available via a FHIR server are: `Patient`, `Observation`, and `Procedure`. FHIR supports three export formats: JSON, XML, and Turtle (RDF). To query a FHIR API for all patients, you can request the Patient resource without any queries. To request a single patient, following the URL `/Patient/id` would access the records of that particular patient.

Studies of interoperability issues. Numerous studies have identified a core set of roadblocks to complete adoption of interoperability standards [1, 7, 16].

(1) Standards may not be multi-purpose: A standard may not apply to all the use cases and may need more support in the form of implementation guides to make it more feasible to use.

(2) Overlap in Standards: The overlap in HL7, ASC X12, and openEHR only leads to confusion regarding an EHR provider choosing which standards to adhere to and support.

(3) Agencies not supporting one of the existing prominent standards: if federal agencies and providers do not adhere to these interoperability standards and require data downloads in their own format, these standardized goals are not serving their purpose.

(4) Constant evolution of standards: newer standards require major retooling and significant re-work. In addition, different HL7 standards have different purposes, and supporting all of them would be needed to support broad scopes of healthcare workflows.

Tools to test compliance of FHIR systems. Table 2 compares the FHIR Garden to other projects to improve interoperability. Although these systems help improve compliance and test the features of FHIR systems, these are not security-focused tools designed to uncover systems' vulnerabilities.

Fuzzing is the process of sending random input through a target program in an attempt to uncover implementation bugs in the software. The FDA had announced that it would be using a fuzzer, Defensics, to uncover security vulnerabilities in software [3]. However, we have not seen any subsequent reports that discuss their findings. Another common direction existing research follows is

Tools	Asbestos[a]	Matchbox[b]	Inferno[c]	Interoperability Land[d]	Touchstone[e]	FHIR Garden[f]
Organization	NIST (US National Institute of Standards and Technology)	ahdis (Switzerland)	ONC (Office of the National Coordinator for Health IT) and MITRE	Interoperability Institute supported by Michigan Health Information Network (MiHIN)	AEGIS	Narf Industries
Open-Source	Yes	Yes	Yes	No	No	Yes
Availability	Free	Free	Free	Freemium	Freemium	Free
Target Users	Healthcare IT Developers and Integrators	Healthcare IT Developers and Integrators	Healthcare IT Developers and Integrators	Healthcare IT Developers and Integrators	Healthcare IT Developers and Integrators	Cybersecurity Researchers, Healthcare IT Developers and Integrators
Technology Stack	Java, Tomcat, Vue	Java (JDK 11), Spring Boot, Apache Maven, Docker, Kubernetes, Angular	Ruby, Docker	Unknown	Unknown	Python, Docker
Scope	Microservice environment for healthcare interoperability standards testing, integrating IHE XDS & FHIR profiles.	Microservice environment for implementing & testing FHIR-based solutions and for mapping healthcare data into HL7 FHIR standards.	Automated tool for creating, executing, and sharing HL7 FHIR Standard API conformance tests.	Sandbox to develop, integrate, and test healthcare interoperability combined with capabilities for complex FHIR implementation testing.	Open access IaaS and TaaS for comprehensive interoperability and conformance testing of health information exchange systems.	Testbench to study the interoperability of EHR and FHIR software tools in healthcare data and the security flaws associated with them.
Features	Supports server, client & interoperability testing.	Validates FHIR Implementations through API or GUI using HL7 Java reference validator.	Checks compliance to FHIR implementation guides.	Generates HL7 FHIR-compatible synthetic patient test with clinically relevant encounters.	Internet-based interoperability FHIR Testing against the HL7 FHIR specifications and standards.	Allows researchers to study the output of different FHIR servers receiving the same patient data.
	Comprises a recording proxy and a test Engine functioning in Server and Client testing modes.	Supports creation of customized Matchbox containers for specific adaptations	Validates individual FHIR Resources to base FHIR specification or specific FHIR profiles using HL7 FHIR Java Validator.	Includes pre-loaded synthetic FHIR Healthcare data and safe harbor PHI de-identification services.	Tests interoperability with other FHIR Server and FHIR Client implementations.	Chains multiple FHIR servers to study interoperability and their import/export operations.

[a]https://github.com/usnistgov/asbestos
[b]https://github.com/ahdis/matchbox/tree/main
[c]https://github.com/inferno-framework
[d]https://interoperabilityinstitute.org/iol/
[e]https://touchstone.aegis.net/touchstone/
[f]https://github.com/narfindustries/digiheals-public

Table 2: A comparison of interoperability tools

fuzzing DICOM (image sharing protocol used in medical settings) implementations [21].

Our use of fuzzing in the FHIR Garden varies from these directions in that we employ *differential fuzzing* to avoid finding vulnerabilities or programming errors in one software but find differences between multiple software that must implement the same protocol or format.

3 Approach

3.1 Goals

We set out to build this FHIR Garden and the tooling around it with specific goals in mind.

- **Comparing Data Structures:** The tooling must allow researchers to study the output of different FHIR servers that receive the same patient data. Differences in interpreting this data can lead to some fields being omitted or overlapping with other fields in the patient data.
- **Enabling Differential Fuzzing:** The tooling must support robust differential fuzzing as an end goal. To enable this differential fuzzing, these engines must support the same input and output types to enable seamless comparison.
- **Chainability:** When a vulnerability is identified, the tooling must enable support to send an EHR through a chain of FHIR servers to study how these vulnerabilities can be leveraged.

Table 3 provides a framework for how we chose our target EHRs and FHIR servers, summarizing their features. Although popular open-source EHRs like OpenEMR and GNU Health provide some preliminary support for FHIR, they do not implement the FHIR Write protocols that allow the creation of new patients using FHIR. Without support for the FHIR Write protocol, we cannot test their Patient import functionality, and they are not fully supported in our HTTP Garden tooling. The table also includes the commercial providers Epic and Cerner to demonstrate how no open-source tooling implements all the features necessary. We also see that SMART-on-FHIR authentication is not commonly supported by FHIR servers, making them vulnerable to releasing private patient data if not secured using other means.

EHR Software	Libraries		Programming Language	HL7 FHIR			HL7 CCDA	
	JSON	XML		Write	Read	SMART	Write	Read
Epic	-	-	-	●	●	●	●	●
Oracle Cerner	-	-	-	●	●	●	●	●
HAPI	FasterXML/Jackson	Javax XML	Java	●	●	○	-	-
IBM FHIR	Jakarta	Jakarta	Java	●	●	○	-	-
Blaze FHIR	FasterXML/Jackson	Javax XML	Clojure	●	●	○	-	-
VistA	M Server	M Server	MUMPS	●	●	○	○	○
OpenMRS	FasterXML/Jackson	Javax XML	Java	●	●	○	○	○
OpenEMR	PHP JSON	SimpleXML	PHP	○	●	●	◐	◐
GNU Health	Python JSON	DifusedXML	Python	○	◐	○	○	○

Table 3: A comparison of the JSON parsing libraries used in open-source software. (●: Feature fully supported; ◐: Feature claimed, but not verified by the FHIR Garden team; ○: Feature not supported). SMART is an authentication protocol implemented on top of FHIR to ensure that only authenticated clients access sensitive resources.

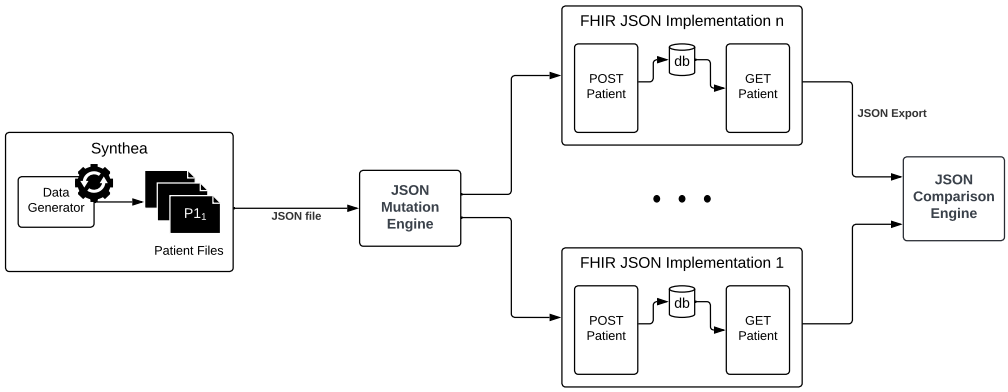

Figure 1: The FHIR Garden approach to comparing various FHIR implementations and their parsers

3.2 Architecture

The FHIR Garden contains a collection of Docker container descriptions and tools to compare their outputs (Figure 1). Using the FHIR Garden, researchers can set up local instances of these FHIR services to compare how they respond to well-formed and malformed FHIR requests. Our tooling was written entirely in Python, and it provides command-line tools for researchers to run files through the FHIR servers and visualize their differences.

3.2.1 Comparing FHIR implementations. For each resource type supported by an instance, we perform the following steps to compare the FHIR implementation:

(1) Get an FHIR JSON file from Synthea, directly user-supplied file, or via the fuzzer output.
(2) Send HTTP POST requests to the FHIR server with the corresponding resource. If successful, extract the created ID for the resource.
(3) Send a GET request to the FHIR server with the user ID and log the JSON output.
(4) Repeat the above steps for all the FHIR servers under test.
(5) Compare the JSON output across all the servers and the original input provided.

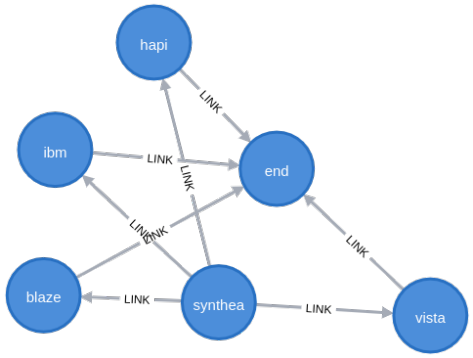

Figure 2: Game of Telephone with data generated from Synthea passing through a single FHIR server (n=1) and finishing at end

Storing the output in a database. We leveraged the Neo4J database to store the output of our "Game of Telephone" tooling and FHIR Garden servers. Figure 2 shows a sample output from our Neo4J database. Each node in the graph represents an FHIR server or the source of the file, whereas the edges contain the exact JSON structure exported by the source of an edge. We also provide

GUID	Chain Links	Diff Score	Diff
7be7c057-6e75-4708-8cee-6a9cb71bc1e6	synthea -> vista and vista -> end	0	JSON FHIR data is identical.

GUID	Chain Links	Diff Score	Diff
7be7c057-6e75-4708-8cee-6a9cb71bc1e6	synthea -> hapi and hapi -> end	0.009	{'dictionary_item_added': [root['meta']['versionId'], root['meta']['lastUpdated'], root['meta']['source']], 'values_changed': {"root['id']": {'new_value': '1', 'old_value': '25dd1a52-3fbd-472f-5352-d72c25963896'}}, 'deep_distance': 0.008968609865470852}

GUID	Chain Links	Diff Score	Diff
7be7c057-6e75-4708-8cee-6a9cb71bc1e6	synthea -> ibm and ibm -> end	0.0068	{'dictionary_item_added': [root['meta']['versionId'], root['meta']['lastUpdated']], 'values_changed': {"root['id']": {'new_value': '190739dd74f-f06d7d7d-34c5-4ff8-9199-fa576a380c88', 'old_value': '25dd1a52-3fbd-472f-5352-d72c25963896'}}, 'deep_distance': 0.006756756756756757}

GUID	Chain Links	Diff Score	Diff
7be7c057-6e75-4708-8cee-6a9cb71bc1e6	synthea -> blaze and blaze -> end	0.0068	{'dictionary_item_added': [root['meta']['versionId'], root['meta']['lastUpdated']], 'values_changed': {"root['id']": {'new_value': 'DEDTTXMY5NCJSOTM', 'old_value': '25dd1a52-3fbd-472f-5352-d72c25963896'}}, 'deep_distance': 0.006756756756756757}

Figure 3: JSON differences between the input and output patient data passing through each FHIR server

a command line tool to differentiate between JSON structures. We used the `deepdiff` library to compare the JSON outputs and compute the edit distance across these versions [8]. Figure 3 shows the output of our command-line tooling.

3.2.2 Using Synthea. Synthea [20] is a publicly available tool to generate synthetic patient data. This tooling provides output in the FHIR formats and C-CDA. Our command-line tooling provides an interface to generate a file using Synthea and use that to compare the FHIR implementations.

3.2.3 Fuzzing. To fuzz the JSON implementations of the FHIR servers, we mutated Synthea-generated FHIR JSON files. Our mutator keeps the overall JSON syntactic structure while applying malformations, such as duplicate fields, missing mandatory keys, and changing the types of various keys. For each input Synthea file, we generate 100 mutations and run them through the FHIR Garden to triage them for deeper analysis.

We run these mutated files through the FHIR Garden and the comparison tools as part of the fuzzing loop. We log the files that produce differences and triage them manually. These files contain minor differences, such as the Patient ID, which may be rewritten from one EHR to another.

3.3 Chaining FHIR Servers

As stated earlier in the section, one of the goals of the FHIR Garden is to support chaining and explore how malformations move through a chain. Our "Game of Telephone" or chaining tool primarily supports two modes of operations. First, we specify a chain, and the tool will attempt to import and export a patient record through that sequence. For example, a chain `[vista, vista, blaze, ibm]` would import and export through VistA twice before importing the exported file through Blaze and finally through IBM.

Second, we support a depth-first search operation where we explore all possible paths up to a certain path. This mode lets us find differentials and when they stop getting imported into systems. Figure 4 demonstrates how our tooling sends POST requests and GET requests sequentially to understand how a specific FHIR server interprets patient data.

4 Findings

As part of our study, we reported three broad categories of findings. First, the support for the Consolidated CDA format in OpenEMR was insufficient, with several vulnerabilities in their implementation. Second, although the FHIR standard adds some additional rules to the JSON format, these additional constraints are often ignored. Finally, we found that there are numerous unauthenticated FHIR servers on the Internet. Several of these servers seem to contain Synthea-generated data or to be hosting honeypots that contain other vulnerable services.

4.1 OpenEMR cannot import CCDA XML files it exported

OpenEMR implements a FHIR REST API and a native REST API. The FHIR REST API, however, does not implement functionality to import FHIR data. Instead, OpenEMR imports and exports C-CDA Continuity of Care Documents (CCDs). These files can be uploaded and exported from the web interface, and the FHIR REST API supports exporting these CCD XML files. We followed these steps: (1) Import a Synthea-generated XML file, (2) Export the CCD file from OpenEMR, and (3) Import the exported file again. However, we observed that OpenEMR would not import these files. OpenEMR would respond with an HTTP 500 error message (Internal Server Error). Upon inspecting the OpenEMR code, we noticed that they use two XML libraries (SimpleXML and xmljson), creating more possibilities for parser differentials.

The OpenEMR team fixed these bugs promptly after our three bug reports on April 13th 2024, April 27th 2024 and May 1st 2024.[3]

[3] https://github.com/openemr/openemr/issues/7417

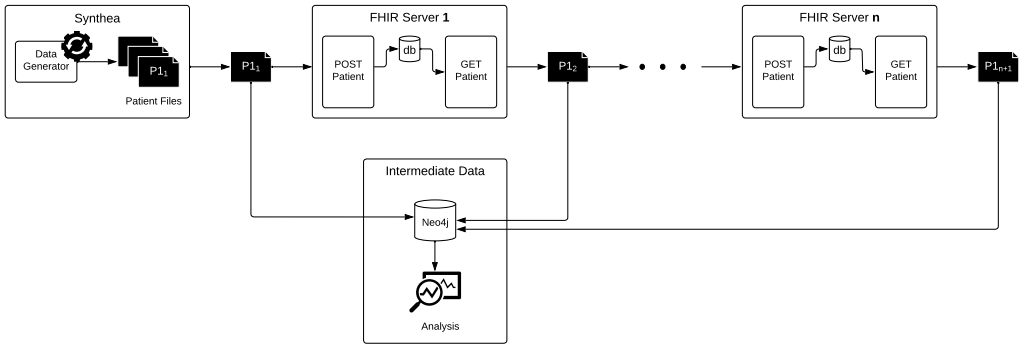

Figure 4: FHIR Garden's Game of Telephone compares patient EHR as it passes through chains of FHIR servers

4.2 Data Integrity not entirely maintained while parsing EHR in OpenEMR

Maintaining the integrity of patient medical health records is especially important while handling them. Subtle differences were observed between the imported patient files stored in OpenEMR, which were identified when importing the Synthea C-CDA XML file to OpenEMR and then exporting the same data. Firstly, time and timezone information was stripped from the timestamps. The DateTime format was simplified to just displaying the date, which could pose an issue for events where the exact time is critical. Next, we observed changes in the vaccine descriptions; for instance, short names of vaccines were used in the original file, and these were transformed to their full names on export. Such transformations could affect the consistency and reliability of patient health records handled within OpenEMR.

4.3 Decimal Value Precision

When we supplied FHIR JSON files containing decimal values, we noticed that the underlying JSON implementations introduced some differences in the exported files. For example, the FHIR specification states the following about decimal values and their precision.[4]

> "The precision of the decimal value has significance: e.g. 0.010 is regarded as different to 0.01, and the original precision should be preserved
> Implementations SHALL handle decimal values in ways that preserve and respect the precision of the value as represented for presentation purposes."

For example, if you consider the input $1e^{+2}$ in Table 4, we see that the Clojure, VistA, and Java's Jakarta and FasterXML parsers interpret and store the value as is, whereas the Python JSON parser converts it to a floating point notation of 100.0, and the PHP parser converts it to an integer (100).

We also found discrepancies in how large integers are handled by FHIR implementations.

> "A signed integer in the range –2,147,483,648.. 2,147,483,647 (32-bit; for larger values, use decimal)"

While most implementations handled it correctly, when we used numbers larger than $1e^{+19}$, OpenEMR at first converted it to this exponent notation, but for values larger, it converted the numbers to an empty string instead.

The FHIR specification explicitly disallows the use of certain special values common in storing decimal values in JSON: INF, -INF, and NaN.

> "For decimal values, the XML special values INF, -INF and NaN are not allowed, and JSON is restricted to the precision limits documented in XML schema for xs:double and xs:decimal icon"

As shown in Table 4, we found that GNU Health's Python JSON parser converts the integer to the value "Infinity," which is prohibited in the JSON and FHIR specifications, whereas OpenEMR and OpenMRS converted them to strings.

4.4 String Encodings

Table 5 shows the differences in string encoding identified across the JSON implementations used in these FHIR servers. We found that most of these differences pertain to interpreting unicode characters differently. The VistA implementation is built using the MUMPS programming language, and the web server is hand-written. The behavior followed by the VistA JSON implementation diverges the most from the others. For example, unescaped unicode characters present a parsing challenge: VistA was the only implementation that did not throw an error when encountering these.

4.5 Syntactic Errors

In addition to the decimal precision errors and string interpretation errors, we also identified differences arising from syntactic errors, primarily in the VistA implementation of the JSON specification. Table 6 discusses these differences in detail.

Bad keywords. Keywords such as, *TRUE*, *tRUTH*, and *trueasdf* are all incorrect and do not map to the "true" boolean field in the JSON specification. However, VistA converts these values to "true" or "trueas" in the last case.

Malformed Arrays. Arrays such as the following, `[1, 2 3 4, 5]` is interpreted as `[1, 4, 5]`, skipping multiple values that are missing preceding commas. Similarly, an array of the form `[truth, NaN, fals, True, nul]` contains all 5 elements that are supposed to be special values. However, all of these values are incorrect. VistA encapsulates each of these values in a string to translate

[4] *NC* used in the tables in this section denote "No Change."

value	Blaze	gnuhealth	HAPI	IBM FHIR	OpenEMR	OpenMRS	VistA
1.000	NC	1.0	NC	NC	1	1.0	NC
1e+2	NC	100.0	NC	NC	100	100.0	NC
1e+100	NC	NC	NC	NC	NC	NC	parse error
1e+1000	NC	Infinity	NC	NC	""	"Infinity"	parse error
1 w/ 19 zeros	NC	NC	NC	NC	1.0e+19	NC	NC
1 w/ 309 zeros	NC	NC	NC	NC	""	NC	NC

Table 4: Integer and floating point encoding differences

value	Blaze	gnuhealth	HAPI	IBM FHIR	OpenEMR	OpenMRS	VistA
"\\u0000"	NC	NC	NC	NC	NC	NC	removal[a]
"é" (2 byte utf-8)	NC	NC	NC	NC	NC	NC	NC
"é" (e + 2 byte overlay ´)	NC	NC	NC	NC	NC	NC	e + garbage[b]
"😀"	NC	NC	escaped unicode	NC	NC	escaped unicode	other[c]
String w/ tab	error	error	error	error	error	error	escape value
Invalid escaped "unicode" value[d]	error	error	error	error	error	error	removal of \u
Unescaped \ in string	error	error	error	error	error	error	removal of \
\uDFAA (second surrogate on it's own)	"?"	error	NC	utf-8 conversion	error	NC	removal
Invalid second surrogate \uD888\u3210	error	"?"[e]	escape, convert[f]	convert to utf-8[g]	error	escape, convert	removal

[a]Removal if first character in string, otherwise it adds a null byte to the string at the same position.
[b]First byte of character followed by \u-escaped second byte
[c]First byte unchanged remaining bytes formatted as separately escaped values.
[d]e.g. "\uqqqq"
[e]First two bytes become "?" then converts remaining bytes into utf-8 encoding
[f]Keep \uD888, encode remainder as utf-8
[g]\xed\xa2\x88\xe3\x88\x90. It is unclear how the first 3 bytes are calculated.

Table 5: Unicode and string encoding differences

value	Blaze	gnuhealth	HAPI	IBM FHIR	OpenEMR	OpenMRS	VistA
characters after object or array	error	error	error	removal	error	removal	removal
Object w/ "," after key & not ":"	error	error	error	error	error	error	convert to key/value
Object with missing key[a]	error	error	error	error	error	error	removal of value[b]
Array with extra commas	error	error	error	error	error	error	removal
Array with missing commas	error	error	error	error	error	error	partial removal[c]
{null, true, false} as key	error	error	error	error	error	error	partial removal[d]
Other non-string value as key	error	error	error	error	error	error	partial removal[e]
Bad value: TRUE	error	error	error	error	error	error	convert to true (boolean)
Bad value: falsE	error	error	error	error	error	error	convert to false (boolean)
Bad value: tRUTH	error	error	error	error	error	error	convert to "trut" string & partial removal[e]
Unquoted asdf as value	error	error	error	error	error	error	partial removal[e]
Unquoted trueasdf as value	error	error	error	error	error	error	convert to "trueas" & partial removal[e]
"String" in single quotes	error	error	error	error	error	error	partial removal[e]
Unclosed object	error	error	error	error	error	error	close object
Unclosed array	error	error	error	error	error	error	close array

[a]e.g., {"a":1, :2, "c":3}
[b]e.g., returns {"a":1, "c": 3} from above example
[c]Removes values up to that proceeding the next comma or end of array. E.g., [1, 2 3, 4] becomes [1, 3, 4]
[d]Parse error if first key is one of these values otherwise key's value assigned to previous key. E.g., {"a": 1, true: 2} becomes {"a": 2}
[e]return everything parsed up to that point, with balanced brackets

Table 6: Syntax errors and differences between FHIR JSON implementations

it to the array `["truth", "NaN", "fals", "True", "nul"]`. No other FHIR server presents these behaviors.

Building an exploit chain. We leveraged the array misinterpretation vulnerability in the VistA system to create a FHIR JSON file where commas were replaced by whitespaces. This change ensures that a file containing a patient's complete record is only partially imported by VistA and can be successfully imported into other FHIR servers with reduced patient records.

Null values as object keys. In addition to arrays, VistA also allows for some malformations in objects. For example, "null", "true", and "false" are keywords in the JSON format. However, these keywords are allowed in the VistA implementation inside JSON objects as keys, but the parser skips over the associated value. For example, an object with the syntax `{"a":1,null:2,"b":3}` is interpreted as `{"a":2,"b":3}`. VistA allows the use of "true" and "false" similarly. This is incorrect behavior. However, we have not been able to show

that this vulnerability can be exploited to affect a patient's EHR inside VistA.

4.6 Unauthenticated FHIR Servers

We found that a large portion of FHIR servers identified on Shodan [18] (Table 1) were accessible without any authentication at all. We searched on Shodan using the search queries "FHIR," "OpenEMR," "HAPI FHIR," and "SmileCDR." Upon closer inspection of the Shodan results of these servers, it is hard to differentiate between Honeypots, servers hosting test data from Synthea, and legitimate patient data accidentally exposed on the Internet. We responsibly disclosed the potential leaks to the government agencies.

The SMART-on-FHIR standard [6, 17, 22] specifies how authentication and layers of security can be added to a FHIR server. This protocol allows a user to register a client with an asymmetric cryptographic key pair. This key pair can then be used to receive an authorization token, which is refreshed with every request. The FHIR Garden GitHub repository contains an implementation of the Smart-on-FHIR protocol to interact with servers that support and enforce it. However, in practice, we found that many open-source servers do not enforce the requirement by default and will respond to public requests for all patient data.

5 Conclusions

This paper presented a novel tool, the FHIR Garden, to allow researchers and healthcare software developers to compare FHIR implementations and find parser differentials at scale. The FHIR Garden can help reduce attack surfaces arising from supporting interoperability protocols and HIEs. In search of interoperability issues, we have identified 59 differences in JSON implementations and other interoperability protocol errors in OpenEMR.

Future Directions. Our FHIR Garden implementation can be further improved in several directions. We wish to pursue coverage-guided fuzzing as a natural next step to find deeper vulnerabilities and differences between implementations [14]. Since FHIR implementations support Turtle and XML formats in addition to JSON, we would also extend the FHIR Garden tooling to compare the outputs of the FHIR implementations across different output formats to search for further parser differentials.

Acknowledgments

The authors thank Dr. Andrew Gettinger, Professor Sean Smith, Ben Kallus, Amish Beg, Seth Leichsenring, Carlos Alvarez, Stuart Nagy-Kato, and Malcolm Schongalla for their help with thoughtful discussions and setting up the FHIR Garden. We would also like to thank the HealthSec workshop reviewers for their suggestions, which have greatly improved this paper.

Funding. This work was performed as part of the ARPA-H DIGIHEALS program under Contract No. SP4701-23-C-0089. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of ARPA-H or the U.S. Government.

References

[1] Eman M. Abounassar, Passent El-Kafrawy, and Ahmed A. Abd El-Latif. 2022. *Security and Interoperability Issues with Internet of Things (IoT) in Healthcare Industry: A Survey.* Springer International Publishing, Cham, 159–189. https://doi.org/10.1007/978-3-030-85428-7_7

[2] Julia Adler-Milstein, Wendi Zhao, Rachel Willard-Grace, Margae Knox, and Kevin Grumbach. 2020. Electronic health records and burnout: time spent on the electronic health record after hours and message volume associated with exhaustion but not with cynicism among primary care clinicians. *Journal of the American Medical Informatics Association* 27, 4 (2020), 531–538.

[3] Steven D Baker. 2014. Fuzzing: a solution chosen by the FDA to investigate detection of software vulnerabilities. *Biomedical instrumentation & technology* 48, s1 (2014), 42–47.

[4] Noah Barsky. 2024. UnitedHealth Paid Hackers $22 Million, Fixes Will Soon Cost Billions. https://www.forbes.com/sites/noahbarsky/2024/04/30/unitedhealths-16-billion-tally-grossly-understates-cyberattack-cost/.

[5] David W Bates and Lipika Samal. 2018. Interoperability: what is it, how can we make it work for clinicians, and how should we measure it in the future? *Health services research* 53, 5 (2018), 3270.

[6] Richard A Bloomfield Jr, Felipe Polo-Wood, Joshua C Mandel, and Kenneth D Mandl. 2017. Opening the Duke electronic health record to apps: implementing SMART on FHIR. *International journal of medical informatics* 99 (2017), 1–10.

[7] Patti Brooks. 2010. Standards and interoperability in healthcare information systems: Current status, problems, and research issues. (2010), 8 pages.

[8] Sep Dehpour. 2024. DeepDiff: Deep Difference and search of any Python object/-data. https://github.com/seperman/deepdiff.

[9] Christine Dymek, Bryan Kim, Genevieve B Melton, Thomas H Payne, Hardeep Singh, and Chun-Ju Hsiao. 2021. Building the evidence-base to reduce electronic health record–related clinician burden. *Journal of the American Medical Informatics Association* 28, 5 (2021), 1057–1061.

[10] Sebastian Garde, Petra Knaup, Evelyn JS Hovenga, and Sam Heard. 2007. Towards semantic interoperability for electronic health records. *Methods of information in medicine* 46, 03 (2007), 332–343.

[11] Ross W Hilliard, Jacqueline Haskell, and Rebekah L Gardner. 2020. Are specific elements of electronic health record use associated with clinician burnout more than others? *Journal of the American Medical Informatics Association* 27, 9 (2020), 1401–1410.

[12] IBM. 2023. Cost of a Data Breach Report. https://www.ibm.com/reports/data-breach.

[13] David C Kaelber and David W Bates. 2007. Health information exchange and patient safety. *Journal of biomedical informatics* 40, 6 (2007), S40–S45.

[14] Ben Kallus, Prashant Anantharaman, Michael Locasto, and Sean W Smith. 2024. The HTTP Garden: Discovering Parsing Vulnerabilities in HTTP/1.1 Implementations by Differential Fuzzing of Request Streams. *arXiv preprint arXiv:2405.17737* (2024).

[15] Edmond Li, Olivia Lounsbury, Jonathan Clarke, Hutan Ashrafian, Ara Darzi, and Ana Luisa Neves. 2024. Patient and caregiver perceptions of electronic health records interoperability in the NHS and its impact on care quality: A focus group study. *medRxiv* 01.30, 24302031 (2024), 2024–01.

[16] Sreenivasan M. and Anu Mary Chacko. 2021. Interoperability issues in EHR systems: Research directions. *Data Analytics in Biomedical Engineering and Healthcare* (2021), 13–28. https://doi.org/10.1016/B978-0-12-819314-3.00002-1

[17] Joshua C Mandel, David A Kreda, Kenneth D Mandl, Isaac S Kohane, and Rachel B Ramoni. 2016. SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. *Journal of the American Medical Informatics Association* 23, 5 (2016), 899–908.

[18] John Matherly. 2015. Complete guide to shodan. *Shodan, LLC (2016-02-25)* 1 (2015).

[19] Siguza. 2020. Psychic Paper. https://blog.siguza.net/psychicpaper/.

[20] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. 2018. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association* 25, 3 (2018), 230–238.

[21] Zhiqiang Wang, Quanqi Li, Qian Liu, Biao Liu, Jianyi Zhang, Tao Yang, and Qixu Liu. 2020. DICOM-Fuzzer: Research on DICOM vulnerability mining based on Fuzzing technology. In *Communications and Networking: 14th EAI International Conference, ChinaCom 2019, Shanghai, China, November 29–December 1, 2019, Proceedings, Part I 14.* Springer, Shanghai, China, 509–524.

[22] Deliya B Wesley, Joseph Blumenthal, Shrenikkumar Shah, Robin A Littlejohn, Zoe Pruitt, Ram Dixit, Chun-Ju Hsiao, Christine Dymek, and Raj M Ratwani. 2021. A novel application of SMART on FHIR architecture for interoperable and scalable integration of patient-reported outcome data with electronic health records. *Journal of the American Medical Informatics Association* 28, 10 (2021), 2220–2225.

[23] Qi Yan, Zheng Jiang, Zachary Harbin, Preston H Tolbert, and Mark G Davies. 2021. Exploring the relationship between electronic health records and provider burnout: a systematic review. *Journal of the American Medical Informatics Association* 28, 5 (2021), 1009–1021.