Source Code Assessment of the CMI Intoxilyzer 9000 Instrument

Drafted For:

State of Georgia

Criminal Justice Coordinating Council

20 September 2013

Principal Investigator:
Andrew Howard
Senior Research Scientist
Georgia Tech Research Institute
andrew.howard@gtri.gatech.edu

Georgia Tech Research Institute
Cyber Technology and Information Security Laboratory

Executive Summary

The State of Georgia Criminal Justice Coordinating Council (CJCC) engaged the Georgia Tech Research Institute (GTRI) to evaluate the Intoxilyzer 9000 device, manufactured by CMI Incorporated. The objective of this study was to review the source code of the Intoxilyzer 9000 for potential security vulnerabilities, non-conformance with product requirements or best practices, and code segments that might lead to an inaccurate reading during a breath alcohol test. CMI hosted GTRI personnel for a source code review of the Intoxilyzer 9000 over three weeks between July 2013 and September 2013.

The findings regarding the Intoxilyzer 9000 and its related source code are:

- Device appears to operate as expected and GTRI identified no practical flaws that inhibit its functionality
- Device appears to appropriately handle breath alcohol values throughout the source code
- Device has minor semantic flaws in source code that should be corrected to adhere to industry best practices for software development but these flaws do not appear to impact the results the device produces
- Device is susceptible to compromise from attacks that require physical access to the device

Herein we provide a more detailed evaluation of the Intoxilyzer 9000 source code produced by CMI. Only the Intoxilyzer 9000 source code identified in Appendix C was evaluated. All evaluations were conducted within CMI facilities and Georgia Tech does not possess any source code. Any previous versions, including but not limited to, the Intoxilyzer 5000 and Intoxilyzer 8000, were not evaluated. Future versions or deviations from examined source code are not inherently covered by this report. The assumptions made herein are valid only for the Intoxilyzer 9000 source code identified in Appendix C.

This study only focused on Intoxilyzer 9000 source code related to transferring calculated breath alcohol data from the device to the user interface. As this was a source code review, this study assumed that breath alcohol values returned by device infrared spectrometry were correct. Additionally, the method and accuracy of the mathematical calculations regarding breath alcohol testing contained in the source code were not within the scope of this study. These calculations require review by experts in infrared spectrometry and breath alcohol. This study does not represent a guarantee that the reviewed Intoxilyzer 9000 source code is secure, will properly operate, or properly handle breath alcohol values, only a best effort attempt at a review.

Evaluation methods and standards were chosen according to the “Technical Guide to Information Security Testing and Assessment” published by the National Institute of Standards and Technology (NIST) and the “CERT C++ Secure Coding Standard” published by the United States Computer Emergency Readiness Team (US-CERT). The characteristics evaluated were chosen based on principles and guidelines set forth in “Software Testing: Fundamental Principles and Essential Knowledge.” This study is only a listing of Intoxilyzer 9000 source code findings not a risk assessment. The risk and potential impact of the findings identified by this report should each be evaluated independently by the State of Georgia and CMI Inc. for potential remediation.

The Intoxilyzer 9000

The Intoxilyzer 9000 provides breath alcohol testing through the use of infrared spectrometry technology. The device operates through an on screen user interface (UI). During subject testing the device takes additional input provided by the subject breathing into a tube. The device, using infrared spectrometry, then analyzes the collected breath sample. The UI guides the user and subject through the testing processing using on-screen prompts.

During this study, the Intoxilyzer 9000, serial number 90-000623, manufactured on 08/2013 was used for testing. Source code review was conducted on three software modules: a user interface module written in the C-sharp (C#) language, a scientific analysis written coded in the C-plus-plus (C++) language, and a device control module written in the C language.

The device contains two processors which are governed by the Windows CE operating system (version 6, 2006). The primary processor runs the user-interface and library modules. The secondary processor runs the proprietary device control module. The two processors communicate via a serial peripheral interface (SPIO).

CMI did not allow the source code or detailed notes pertaining to the source code to leave their facility. Given this limitation, this report only contains high-level observations from the source code review. Appendix C contains additional information necessary to identify the reviewed source code.

Georgia Tech

The Georgia Institute of Technology (Georgia Tech) is one of the nation's preeminent public technological universities, ranking first in the nation for Research and Development at a public university. Georgia Tech maintains a robust presence in the information security research and development field and is designated as a National Security Agency Center for Academic Excellence in Research. Georgia Tech's multi-dimensional organizational structure allows for agile research in both theoretical and applied spaces.

The Georgia Tech Research institute (GTRI) is the applied research unit of Georgia Tech. For the past 75 years, GTRI has been the non-profit engineering and applied research and development arm of Georgia Tech. GTRI has extensive capabilities and experience in the design, testing, and sustained engineering for technology systems. GTRI brings a key attribute to the management and development of programs, namely independence. Because of GTRI's not-for-profit status and because GTRI is not a manufacturer, GTRI solutions are innovative, affordable, and in the best interest of the customer.

Source Code Review Objectives

The objective of this project was to review the source code of the Intoxilyzer 9000 for potential security vulnerabilities, non-conformance with product requirements or best practices, and code segments that might lead to an inaccurate reading during a breath alcohol test. Only the Intoxilyzer 9000 was reviewed.

The assessment examined the source code for the characteristics listed in Table 1 in addition to any obvious source code errors. The characteristics were chosen based on principles and guidelines set forth in "Software Testing: Fundamental Principles and Essential Knowledge."

Table 1: Source Code Characteristics

Characteristic	Description
Casting and conversion errors	Statements that inappropriately convert the result between types such as converting floating point numbers to integers.
Mathematical precision	Calculations that cause precision loss due to operations on large numbers.
Mathematical range errors	Statements which inappropriately use math functions with defined ranges (e.g. square root function).
Mathematical exceptions	Division by zero or operations on NaNs (not-a-number).
Input validation	Validating untrusted inputs are properly checked and formatted for use within code.
Sanity checks	Examining code pathways and functions to ensure that pre-conditions are met.
Bounds checks	Examining loop constructs, arrays, and other collections for conditions that may operate outside the boundaries of the specific resource.
Password handling	Checking that passwords were properly stored and handled.
Encryption algorithms	Checking for insecure (or not proven secure) encryption, checking for

	proven secure encryption (AES, 3DES, etc.) being used in accordance with secure guidelines.

Testing Methods and Approach

Overview

The overall assessment was conducted according to guidelines set forth by the following agencies: the National Institute of Standards and Technology (NIST), the United States Computer Emergency Readiness Team (US-CERT), Mitre (MITRE), and the National Vulnerability Database (NVD). During the evaluation the standards set forth by these institutions were the guiding factors in determining the severity of potential weakness within the code. The referenced publications can be found in Appendix B: Bibliography.

Throughout this review the reader will encounter Common Weakness Enumeration (CWE) references. These references were chosen by researchers to represent the weakness or flaw that the code exhibited during that phase of evaluation. CWEs "provide a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design" [1].

Assessment activities were conducted using a three-step approach:

Step 1: Source Code Familiarization

GTRI first familiarized themselves with the source code and the architecture of the Intoxilyzer 9000. During this process, GTRI attempted to understand the organization of the source code and how different pieces of the source code interoperate. GTRI identified relevant source code files and segments for further analysis.

Step 2: Automated Review

GTRI then conducted an automated review of the source code. Automated review is a method of reviewing source code through the use of automated tools. GTRI utilized commercial-off-the-shelf (COTS) tools to identify potential software vulnerabilities in an automated fashion. The tools which were selected are noted in the Source Code Assessment section. The advantage of automated tools is that they can review a large amount of source code quickly; however, false-positives are common. After automated tools identified potential issues, a human review was conducted.

Step 3: Manual Review

In the final review step, GTRI reviewed relevant aspects of the source code manually. During this step, GTRI researchers walked through each line of code that handles breath alcohol values. This review did not include a validation of the methods or calculations used to obtain breath alcohol values.

Physical Evaluation

The device was tested using both standard usage and abusive usage.

Under standard use, the device was operated in accordance with its operating parameters, including using approved peripherals such as USB keyboards, USB network, and Ethernet connections. During this testing, on-screen instructions were followed when operating the device. Primarily, instructions presented for using the breathing apparatus were followed to get a valid result.

Under abusive use, the software was subject to any (non physical) means necessary to coerce an incorrect result. This includes overflow attacks on input fields, improper data formats, attempting to inject SQL commands through various input fields, connecting both approved and unapproved peripherals, and using the breathing apparatus in a way that was inconsistent with normal breathing (or using external air sources).

Under both testing scenarios, connectivity tests were limited to plug-and-play (PnP) functionality common to many operating systems.

Source Code Assessment

Software vulnerability assessment consists of examining code, especially data input pathways, where that data can be used in a malicious way to coerce the code to perform incorrectly. Per the methodology, automated software tools including CppLint and FxCop were used to generate an analysis of the code. CppLint is a C/C++ static analysis tool which analyzes code for both style errors (poorly named variables, poorly indented code, poorly scoped variables, etc.) and for typical coding mistakes (variables used before initialization, buffer overflows, memory leaks, etc.). FxCop is a similar tool for the C# language; it performs both static and dynamic analysis of source code and will report similar errors to that of CppLint. The results of the investigation included herein contain results obtained from these tools and verified by GTRI researchers.

Semantic assessment consisted of a general overview of language constructs to determine if subtle flaws existed that are commonly overlooked. These flaws exist because the code may express the intent or idea correctly but the implementation cannot work due to physical limitations of the hardware, inaccurate representation of the semantic construct, or an overlooked flaw. For example absolute value functions fail when converting maximum negative integers to positive integers; the negative range contains one more number that cannot be properly reflected on the positive range.

Security assessment consisted of examining protocols surrounding access to critical device functions. This included passwords and access to administrative functions, encryption schemes used for data handling, and network protocols used for updating firmware or download information from the device. This assessment is to ensure that proper handling of sensitive information and proper protection of critical functions are employed.

Findings

During the source code review, six *potential* vulnerabilities were identified. The findings indicate a potential only—no finding led to a successful exploitation (adverse manipulation of the device or its output). This does not indicate the device is free from *all* exploitation. These findings included possibilities for SQL injection, floating point exception, and buffer overflows.

(Finding 1) SQL Injection: More than ten statements had improper handling of SQL variables and as such could possibly be allowed to contain data that could be misinterpreted by an SQL interpreter thus causing an exploit. In practice, all attempts to perform actual SQL injection were unsuccessful due to input field constraints such as limiting the number of characters allowed in the input data or validating that the data was in a specific format (i.e. date fields were entered in a MM/DD/YYYY format). (CWE-20, CWE-95)

† CWE-20: Improper Input Validation (The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program.)

† CWE-95: Improper Neutralization of Directives in Dynamically Evaluated Code (The software receives input from an upstream component, but it does not neutralize or incorrectly neutralizes code syntax before using the input in a dynamic evaluation call.)

(Finding 2) Buffer Overflow: At least two buffer overflow conditions were identified within the code. The specific flaws were deemed non-critical and were not accessible from the user interface. The non-criticality of these flaws were determined based on the type of resources utilized during the overflow (constant strings), the static memory layout of the device code, and that no adverse runtime affects were observed as a result of these overflows. (CWE-787, CWE-805)

† CWE-787: Out-of-bounds Write (The software writes data past the end, or before the beginning, of the intended buffer.)
† CWE-805: Buffer Access with Incorrect Length Value (The software uses a sequential operation to read or write a buffer, but it uses an incorrect length value that causes it to access memory that is outside of the bounds of the buffer.)

(Finding 3) Precision Loss: There were at least ten statements that could *potentially* lead to a round-off error during floating-point calculations. Numerous statements were identified within the analysis routines that used floating-point calculations that could result in a loss of precision. These calculations relied on the implicit assumption of valid numbers and/or numbers that would result in a valid number after the calculation is complete. Based on review, calculations throughout the source code preserve at least 8 decimals of precision under normal operating conditions. It is highly unlikely that this precision loss will impact any breath alcohol calculations. (CWE-349, CWE-681, CWE-739, CWE-885)

† CWE-349: Acceptance of Extraneous Untrusted Data With Trusted Data (The software, when processing trusted data, accepts any untrusted data that is also included with the trusted data, treating the untrusted data as if it were trusted.)
† CWE-681: Incorrect Conversion between Numeric Types (When converting from one data type to another, such as long to integer, data can be omitted or translated in a way that produces unexpected values. If the resulting values are used in a sensitive context, then dangerous behaviors may occur.)
† CWE-739: CERT C Secure Coding Section 05 - Floating Point (Weaknesses in this category are related to rules in the floating point section of the CERT C Secure Coding Standard.)
† CWE-885: SFP Cluster: Risky Values (This category identifies Software Fault Patterns (SFPs) within the Risky Values cluster.)

(Finding 4) Integer Overflow: There were statements that could *potentially* lead to an overflow error. More than ten calculations were witnessed which multiplied two or more integers and stored the result in a similar sized integer. Depending on the input of the two integers the resulting calculation may produce a number larger than what the result variable can store properly. During testing there were no numbers witnessed that would be this large to cause such an overflow; however, such conditions could not be eliminated as a possibility. Additionally, it is likely that an integer overflow will cause the device to error. (CWE-190)

† CWE-190: Integer Overflow or Wraparound (The software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the calculation is used for resource management or execution control.)

(Finding 5) Sign/Unsigned Conversion: There was a superfluous use of casting signed integers to unsigned integers. The code made assumptions about the signed-ness of the integers without actually verifying that the variable was positive. During review, there were no negative numbers witnessed; however, such conditions could not be eliminated as a possibility. (CWE-195)

† CWE-195: Signed to Unsigned Conversion Error (A signed-to-unsigned conversion error takes place when a signed primitive is used as an unsigned value, usually as a size variable.)

(Finding 6) Weak Authentication: Users are authenticated by a user identifier and a short password. While passwords appear to be stored securely, the device uses a relatively weak authentication scheme that can be brute-forced or guessed. A privileged user account, who has access to all of the configuration pages of the device and can alter many of the fine aspects of the test, is not required to pass any additional authentication challenges, and can be compromised in the same manner as an unprivileged user.

By pressing a specific character sequence an operator can attempt to escape "Kiosk Mode" and break out of the Intoxilyzer 9000 application and into the Windows CE operating system. This requires a short password that is computed in a predictable fashion. Not only is the password predictable, but also a short password can be brute-forced. After escaping "Kiosk Mode", the operator has privileged access to the software executables and database contents. (CWE-521)

† CWE-521: Weak Password Requirements (The product does not require that users should have strong passwords, which makes it easier for attackers to compromise user accounts.)

Appendix

Appendix A: Cobra Analysis

The State of Georgia requested that GTRI review the Intoxilyzer 9000 network communication source code. The network communication capability of the Intoxilyzer 9000 is referred to as Cobra. The reviewed Intoxilyzer 9000 configuration for the State of Georgia did not have the Cobra functionality in use; however, the source code was available for review. As such, the following findings do not specifically apply to this study but are provided as requested.

(Cobra-1) Firmware Update: The device was found to have network based weakness when the device is used to update its firmware. During the update an encrypted zip file is transferred to the device and its contents are written, unverified, to the flash memory device. An attacker could decipher the proper filename and location of the executable and craft an updated zip file which replaces the current application. If an attack were properly executed, the application on the device could be overwritten with a malicious application. (CWE-327, CWE-347)

† CWE-327: Use of a Broken or Risky Cryptographic Algorithm (The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the exposure of sensitive information.)
† CWE-347: Improper Verification of Cryptographic Signature (The software does not verify, or incorrectly verifies, the cryptographic signature for data.)

(Cobra-2) Encryption: The device uses an unproven, breakable, encryption scheme to encrypt all network communication to and from the device. (CWE-326, CWE-327)

† CWE-326: Inadequate Encryption Strength (The software stores or transmits sensitive data using an encryption scheme that is theoretically sound, but is not strong enough for the level of protection required.)
† CWE-327: Use of a Broken or Risky Cryptographic Algorithm (The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the exposure of sensitive information.)

Appendix B: Bibliography

1. "CERT C Secure Coding Standard." *CERT C Secure Coding Standards*. N.p., n.d. Web. 05 Sept. 2013. <https://www.securecoding.cert.org/confluence/display/seccode/CERT C Secure Coding Standard>.
2. “Common Weakness Enumeration.” N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/ >
3. “Common Weakness Enumeration.” *CWE-20: Improper Input Validation (2.5).* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/20.html>
4. “Common Weakness Enumeration.” *CWE-95: Improper Neutralization of Directives in Dynamically Evaluated Code.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/95.html>
5. “Common Weakness Enumeration.” *CWE-190: Integer Overflow or Wraparound.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/190.html>
6. “Common Weakness Enumeration.” *CWE-195: Signed to Unsigned Conversion Error.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/195.html>
7. “Common Weakness Enumeration.” *CWE-326: Inadequate Encryption Strength.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/326.html>
8. “Common Weakness Enumeration.” *CWE-327: Use of a Broken or Risky Cryptographic Algorithm.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/327.html>
9. “Common Weakness Enumeration.” *CWE-347: Improper Verification of Cryptographic Signature.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/347.html>
10. “Common Weakness Enumeration.” *CWE-349: Acceptance of Extraneous Untrusted Data With Trusted Data.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/349.html>

11. "Common Weakness Enumeration." *CWE-521: Weak Password Requirements.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/521.html>
12. "Common Weakness Enumeration." *CWE-681: Incorrect Conversion between Numeric Types.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/681.html>
13. "Common Weakness Enumeration." *CWE-739: CERT C Secure Coding Section 05 - Floating Point (FLP) .* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/739.html>
14. "Common Weakness Enumeration." *CWE-787: Out-of-bounds Write.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/787.html>
15. "Common Weakness Enumeration." *CWE-805: Buffer Access with Incorrect Length Value.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/805.html>
16. "Common Weakness Enumeration." *CWE-885: SFP Cluster: Risky Values.* N.p., n.d. Web. 04 Sept. 2013. <http://cwe.mitre.org/data/definitions/885.html>
17. McCraffrey, James D. *Software Testing: Fundamental Principles and Essential Knowledge*. BookSurge Publishing, 2009. Print.
18. "Security Management & Assurance." *National Institute of Standards and Technology*. N.p., n.d. Web. 05 Sept. 2013. <http://csrc.nist.gov/groups/SMA/index.html>.
19. "Technical Guide to Information Security Testing and Assessment." *National Institute of Standards and Technology.* PDF File. Web. 05 Sept. 2013. <http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf>

Appendix C: Source Code Reference

The secure hash algorithm (SHA) family was chosen based on NIST recommendations and version 1 (SHA-1) was chosen based on the available tool set and installed components on the CMI-provided testing machines. Hashes are used to identify whether text has been altered—the inherent property of hash calculations are that the same identical text will produce the same hash value, alterations of the text (even minor) will produce significantly different hashes. The odds of a duplicate hash value being produced by two different texts is about 1 in 10 quadrillion (20 decimal digits of precision). The following hashes were taken from the source code and can be used to identify that such source has not been modified since this review. CMI did not allow source code or source code filenames to leave their facility.

SHA-1 Hash (File Contents)	**SHA-1 Hash (File Name)**
d0323b00a6c868af6a27a584ceae0e033a3fe04c	0ee4b97aecbf3a580ede08884c490214b7b27164
23284b57b49badabc1d30ea79f35c9a95549793a	1cf33db41c1351d5dfe6bc89e182b5b708741aa3
6768b1119484567e4b5cb60859bb25188e630cf9	1f06dd04e60e7328c88211159ef430c7f93f40fa
5f0a411cae35990d1baa1159bd303e6d385f5de4	f21b0df03de4b1b8b53077ba8ac76cbf30289d95
b4d689fe726694d9b71b583b8f01c778afb31d86	13304e53a7009a036f0faf28fe1e1f674aa57214
7e53553a5f64785b3ef4ade472bd8bde37303878	75142544d1d47b68be9b6c104662827d05afe3b8
5886fd712d38ae6a3dc343bb69fa0fd931f8acc1	eee4238c7ff184ca0dd02b194679688eb560e0d3
1a96bc407a975cfb465e43d3c6aad6bfffdde1aa	15ed24a3f8d0dc37f9f821ef1ee651b96b73680c
f86ca2dc45d40a5664a11f14d442bf877866e6b8	dcd6c3bb4227b1dabb8fef100d7e3e6916b27631
54dbf32ab1ab054f726db8e6c2081cbffd58d4cc	bddb2b729393704f8ed7074a508118f7e699656c
0fbd5c4fbd181913b84f1e892ccc9ca3d082b8a0	b4dc454146f299f69b57faa9bba599e3185843dc
5de929946563818deff17f79c310a494a0b5c2dc	bf6ea22b282c99e3068f77bf7e502530d0dbcbae
64cd411443340f29e119b327737f23261d4079ea	980c806f3a85e0ac5806782909f087363975e094
5c0720d2ad66fb189d96e029d43af75e8e5d2179	d53e40d2c3c4efbb034f8b28c0f3b728598ce3ce
1b51a4a8839c1e743e17170ecf150b6eeb3b4084	562bef3d112f4bc8e1f9f0c98462d8bcfa1b9240
77b30484b2494b85d33af7f0882b5ff7251782ac	d913cec3fc2a1bd9029c92e66710f9f84493d79b

4bb346498e98039af643c19ad972a420804610d5	a3dc945bdd988d964d4aca769649889a26fb7470
ed61f55d7cfed38840f5cd1d039349e8fe3f4ee1	ff2adf48a040cc6276242df22a52f5de33544171
caa2bde7197db3aeeb0cf1e8feaf65b8d99c0fab	deaed384361d8aa283387505bca6ca8386980095
d721ad9d1b323e7f22987f7294817acf1251b3ff	77876f2d9d113b56a62eb57a87055c583b226a30
b6aeb553ed3ed91b2e6b890d70f42e7c40a27e7a	004f54adfd3a314be3fb6105b79bbe9280ac7ecb
7a516e06715caa87df64498d1242b887a8953fc2	360fecf3da3410272ff220e0b47d06978a508870
c7646b7fc3e8d28fe1576443152088e147563e0c	29cff5e835cf9ed4baf3d37d8315f8cd3cd4b75e
4004ac88ee3e9349c856883bfb07ab0673cb1284	d5663ad8c8473efb2b3970b6f2af27a4cb58333b
841704625cfbf635c765ca7e8519fa3492a001f2	5587a39ff62a79193f5ce81a116c02102804e222
a2d856201cefea147a649528f9269c993293d538	507c5a869b02c01de613a84e9cb50aca2addb45e
315b86a796d84f3a49e9960db7655716493477e2	646b55c45f272bea42780aea91dc3855a2f087d0
9811bb39118fed5b4d2e708315b5998ee98191df	085f5e0b25c284d4f2e56161a1aa4c1aa8798ee8
9d3e521af3e1c76a2b58fffadb35d827342f1b82	07f3ede0b7d6f08e0149cd269f9820808879e8cc
9ff3f5364fad747acc4537f05b1f249f4955d564	816802cfa9837e0dd1daf981a5ec18333a06a8f0
541514a4ce7b112492b1ac367466fa528ff90f2b	2466f7608e1e252588763dadcb2df7d56f57636a
c798aadedf6b5a00235a6f6c5a669805173ddfdd	075e26f8d0174dcc4187048319bda58afba8bc16
00677b8e81eb2f0c1d853aeeea80aeb224ab53ae	94283539a9476c684bc7045febb755704e322099
49e9058db54d0797cef0485cceb28ccee960d54e	cf7b17efebced6c7c4050ef22a3aa6b6e1227ae0
01453fa840ff4c6a0bc1fd308b07bb9fb2da1d20	a544f01828d1206b4ad23ec8f0c151c30df8e532
51c8322db6e441d6eb9e10c8bf69780991c45945	4340021a7a00126e37ba381ca0cf3f6f16e2a5e7
b67eaacd1e5d0563bb825a7e64bc9d0cf76d32b0	6012c91f37678a1de88050bd8267c508911bc128
9b0884be57c7bb2e05e1cc2db03d9b2f161c5f52	901d76c4b370ebdd15a7ca62008a3b62a5d3a339
f2dad6b1679849aff18f01644ad70d68f7358915	4caef687ada58ebf13e4e177ab3dedc5a50570fe
25804f25390ff755473e12341154909e8ca1c0d3	687aedc54fa0e006d23488d04c24d661d6ccb516
186a9d86799e4f4c1762e6dc2a40509b910d754d	2e769741606c24f0e0098e15039b9d1374d204b0
70ba5e8ab85af21898ba6b0f99b71d66969402e8	2c864e82976ab01ef755322686785b46e44d1a38
0c83641c1dd2b68fcddca0b1f2d5187c4a0b4896	385b173b54b5e1334c26894c8b369325da254943
caa2bde7197db3aeeb0cf1e8feaf65b8d99c0fab	7b2c61e9fa0d4ebc4c6f5d56cf6f466a4f27442d
6301ecf99acfc386ba2691a2baddf57f9529ee61	c7afb59a59b8c3658d668233182298b6214bb9f1
ba9754d3157da31850c1b5965cd7181cb9db7da5	51c9cc30ba641e341d412e89fffc2a5cca7915d4
0eda67475d680c8cab6281ba129e21ee3cc98d44	8d397b0d96536e730d9e47abe05c0e29bb71982c
f46fb9ca663c1c29150258eee75bba15890030bf	9e6bb6e9d75808ffef2da964cfc29e1eb4d3e542
dfb654fc2bc6753ed6a9b9b017a8d945a5114c11	04da5549b6558f61689473657a83014b9a4d7f79
9d5f08b0006b1b2535bc95e9947f9ca347c9c826	2ae21586df47fce55bc6a29e1318c8580c5914ea
e1036330d20b7a78ff8ec2302eaee4133b9baa6e	e5c35b885b6e93505c197b9c0ae5d73fa5e5fe30
c2f09e1119885f760d57cb9a491aa8f93f47aa3e	5dea1f3d71b5defe9cc0feda5579e77303b890db
3289efc9ef219aeb65a0eae7bdcb8c8ef2e284c2	3370dd479d075f601c7838988e67e448146559d6
ae3f280689a8199365e7af17e4cbe929fe44f4b6	95b9e427cc16ca34293f9ef5438b32e8a895fef1
cf645916011fa48751d8a3bd9ae805149e265bed	f9424e6085f99478febfccc0df85fa6c58b463f0
2aa8b3516682794b53fb8a134452c5a6c3036ca2	fa0a342e962ddf51b6fbf1db800b21a0f63e1929
d9f533b3f59052b6ce70fafa8499455b463d55b7	c69aca0b2bcce1163d11f7107c4a5800d4de62e6
ba6cdeaf151bc4a2daaba4b49ed0bea311ee37a3	1346e2b5633f6b2033c1c2075e10a898cbc296f2
6cf2c03e4e998ae070db86c284c7bd15a9d5a491	c0d299ecbe2532c3b510d8fbb58d42f333dd6e71
1de883fc6c3ea9770e85fd2b1bcbaed8412428a0	762c50a558bd0d8c93567bd218ee83739741d71c

7a67cc914577fd1b20f3d2977c14709508aa673a	54ec3005498dd97401a6936d64b9751a9ce2ac36
84c7de4fce6065ebe149f145869859321c0375a5	0dfbb665fff6bd9f7b589e80f9e010486f68bc30
2f45d0b2320c8ed0bb19f22c71636f5149005518	93b474fdd7eae00e816d8f5be04e7e69b3cedf34
b06b30c6a47c6186a5591f1833c9f1e3af7fa8e6	ae18acdbaaf9e24c4a2e67671a0a3201d64288f5
caa2bde7197db3aeeb0cf1e8feaf65b8d99c0fab	31f59528fb6ddb094fb3284c9968e23106266dbf
bd38d11dc1b26a43a62d2f4f9e7cb4b694970b01	758a3ae66a97243883b1c6dd60416561c3f7fa78
97756eaee5a4cc00864cb8856b6469cff9a73756	b87009bb7340b287d897f70940f69804225f1264
ef9b2655b5b45c5828351f66aaf023af97ece203	b54b485cae1639048db3a91ff6553f35c5e12f1a
de2a24716cdd6001feca7043ac16c3c64d43d2db	03fc6204e36e51585b1d9b0cba4bd48b49dae47a
6b6b2e79d73034b407e0b22f2c979ceac8e6c713	2dec2196959a5dfb99fafa2571dbb3b3ca5bdd27
ae52e02b2b6204fb223447e7027540adc4f53d1f	26860b675c9840162566bd4439efb2ab0b31eb08
d45ebae0e8c366a723f84eccefa18349ce27ce4d	a8a3bc9a7ee80e2dd8f273cdfa1875b196bd756c
fc925be1a93a6fd1195c2c84a7df9f36e98b5857	56d51f90b98d353af1e1e1896f5280d8e37ffe58
b9c39e24fdda962116577b6e26af6ff468c9af09	8f2a1ea5d89b671194e6cbeccd9cfccda5cb8029
5301e0681c1b4cf939f6aebcbafdf63caa8a3e85	2f21e30c5addd5cd249c226663ccb9012d2894a0
1e5375b06d26a7ba4c30317cbc340c7be68c2118	af119ed30c2f66f03a960e5248760e587c2e7817
23b53b7c42b20d0963ebc04e92fd51ed3f71b077	12a29cdb13cc250023e238d271b4c675ee29e9b7
dacb2c6642edaca69c68c64fdb94efca2c1b698c	5b6d8e6fe0e9d79b26c84b1f3d01c2769e9f6b1b
bd7e49b95a1479fedd8b77b0db3011c120a5017f	da4c8595ae4ab7b4789b09d053657fb74519bd7b
10666d1d30a556907dd01dd3fe046654852f7167	3c9023bca13292c9393ff1db08467c6cf510c772
f18f384a6f8a58df1bf883e35a08369b84fe3422	651a3f7fe049987c6379ad90a98e0d02fc5f7265
99ff5e1ac842ecaaabc1289211c3d91c3865f92d	76ce4b24c825938458d90a542663a1ae3a8dfa16
cad703294ae549bcae46c75d59c904287063ad1f	0799227ee81bb320828ed38dcac55fb82aee8615
64ca02ee684085f67adc7d727eed0f66c331d30e	c964c3bf808cf5c0c27476832948a25cc7c450bd
beec717c9dcdf123718335634edb1c57ee35da97	b1a494b4f3d646a2391b5c6332be3ba3491ffb24
fba33629acbee413493c10e1f4ba61857b86a821	97270bdc017ab6a7bb64386648fcfe983cc82768
307dcff0b716a60deceeba6f289162a28d8b7202	1fa30d2de4a26de58b7392adcd0cdba4c9d4a442
af7665085f01ce27bbb9cf3db6e3a084eccd86c6	9b52691be79beac87ae7741a9e0a378f84e0d2cc
1e20a5fc316d6ec91664cb951489bfb54e585afc	d55df2c17c2f3a9c3ede14de273a80b8a15be19e
caa2bde7197db3aeeb0cf1e8feaf65b8d99c0fab	4d799f67b6556efb5b54d07782e39058fa7b1ec7
9094f6c204038c9655df93d1db53a09b53a30cdf	3b44f4c15bfc4994f3f534bb10cd0b9bdaff889a
b0c4d91dad7c782e3ef1ebfc366ffec1360d3213	fa4c0b565f21b57cb98d61c86a962e59639953c6
5bd455cf6039ebb0b391152c040dc54833f7280c	45260bca89e408e4e03f1a58b1b19318ece41443
027f5d9cabbd1cb97e1cee4005dce30902a8a2ae	cf6e47a350b03b47a669b41aad69b583d1885760
c8bf784273b7bfa09caa04510f0fc8f9a8c3200e	03818484ec3c95c92dff0ddc2d659ed000774bab
5144321eb99d855925265dec0ff6c3b78d86e023	194d07d2acaaa9600edd5c979130bd30db101ab5
6cba87701748ce63f1317f3c33580bbb98f920c5	25264d2ad94f4818c6ff9c0dae9419c222d57633
65021882eb78aaad7cbbd69980050938131ff22e	d0799c80b38094b94a5fb5673b529f42d007e905
4dc00ea6168778a6100139f9b156d2b55403419b	4110b3b2d4d56b42aacf95910ebd33d05d0708b7
e6930f448b474b57864c413ba98ca632870dbb7a	3e1a9b85b2e6d0ce4d17550bdd11fc8c71e9fea9
bbabde679886244ebb11ac8fd07fe7ebeed0b6b5	e1f1135e179b4f98b809bd848254e342d1256406
aa7ec2f61cb938cc41a95ff2aed62439d67a5ad4	c684bf35f09bc588923c239add1b2a3a6ae42147
0b19164963aa8db36c12bd9f6a562e4536d70f2f	37324c66cb1c8741722383b11832b5982f8ef7b4
1661f73d0879870194f0adc432310a535ab2d8ba	82a5dc187e7a9a9a8ca8c77c3aea5953b7cc5e68

3df4347e39dc45fceb2b8e70d368fce0a3b41e9b	06f4abe0b12253238a1f41e0b446ada95eefee27
3c39d9cffd98a9a45332ac06008c236e038d1625	b0afa50845e2fe273757d6a00532835eae5ad80a
679a5decfad0256b7972d185c8aa707219084ec1	6a347d6c80468327845e59d12aba9b5467d83df5
55f682ae00602d0cca1837bab23924df6088935e	9a8fcb0d8f774b642004f06e0d65590f002ff643
aeeb907b5af311a6d83331016b9c6f70b04f2d67	1456e2e0ea7662c034dd088ab02852975594d335
fe69f57a7abd74429a2c57f6e2a744f33d7dc84c	82c52fa4fdd249daf9eaae1266d0e37ef1b464e8
46c861bd6e24d3addc3cd6eb1823cc6888fecb74	421a908f005b4f1bfcf346363f1686a302c9e802
32a79a0ba1c5e6cca4d5caa69bf112ee682e0572	65b773f9611e9529f98bafa34ea800cf97967363
64ae95b710f870e74ab04d016bfab8a4c5704709	dba5f941b2a9553ad4a84626acfbc4406a9d3cea
2a3e20c5b1e02f1ce052114a9af972521a310ee2	59a3a0b54cf4f1e63706ebcdf05b1a1e0c58b59b
caa2bde7197db3aeeb0cf1e8feaf65b8d99c0fab	f2c062f1765c378ef0983a0ef3693ddc6ed096ca
3dbac59cc9c3db9a8a6f9ad5bf6c208f8e263a00	79a4c5d8187d05006f0eeecdc7f59eebacdf85e7
86a56648d5b49d145404304af8f4395b3c8dcdc4	973234531a591b3cfeb3d10b4f6a526d31c41971
c89dab5f05bfb3db1daf422e8f61563a756ee331	d10d2497fed0fc2e160b86fb5d70bfa6caf4ec21
c75cf2d10fd727b3367c7d9b2332fb14dd1fcdc1	8910b1ab2028baf8af902df35516e0cd9e238b78
ac00196a017eae31de167deff0df04c5afce16a2	e3a8a0902eea367815b1541ea4070fd4215e167f
209659dda1ec5a0b470519e9e21879f761b46d93	e48f2aabd51665690d3b9d0cfc96afb6b533b543
a6dee2b9d641942c73d31a7864a79694d510b496	1328899a60c1d05b6f9395baa312fe71aea6bbba
29f7cb07f436cf22f6388272f0af4cd486757640	69e8a18fd065d1b85105f3975141d731801968b0
bddaf3d65e56522a5149b4dcec7c0461a50b0005	311d39c030c655231e6b62d61f41f90706b04ca1
8b2f67383483845736b6645a548f4d4dd43fa29d	b0386fdc8df96d7e5520d314616f1cd64c00b5c6
5f04b98be73d460754e44c6f74321700ee6cbc80	e3a383710dc445555de678754eb562799a99c4fa
9313c259a3d1314579c63a5f6aab51d50bf6e410	5ded514b47f6f34d83fbd7382e311a29e86c2a47
7af9bdcbd3e924b1d83fbe16683a7feca8609f33	69c4830d523cc85f946fcb62cdef87cbeee9c21a
7f0b6087236f42171212ade6030732ff1f269cbc	e6d22d919365bd0fe871b08e48745ac3747b4e03
a4ce2b3c04f0dd5ad3886aa37f41e0015f06d8e5	fa891d95d4b44abcc587a45914490407e2df4c8c
51594fb8c24af246a8815fb740d67267798259dc	5d8c97df23f6608709acc118de9123eeb18f7747
aba73fb1c87b938ede7400a8c3443161bacf031d	8f319a1656c2da5fc5a14c9d1c81bd7ccf12f65b
d1029eb06f03b49d789f1bac5ad810a92821689f	1a70816c630f9f9ebe3436b84f314510695aa05e
fd9983f9df113393fd5e91a73bdf21ba0e5afcfb	f9517ecf28f0bd80cae2d2c22b8b0b05efbd9357
de50858b2d4c7d3eb6575b505346ce98bb1ab246	daf0fce66ba918a0f8c48e52ed73b4569c4db059
c996ceddef63bb6792bde7b54c63c727ac0f4527	26a061d51e3ef0c49fe59fc0bb54a93202c96931
f0a5fd3b0390128a3beceecec291680bdee15bdb	07f53895ea4d08c1ef6a27b04e7e93878ab4bfda
3887a7b4b31008197ef264a11b160623289c93b4	f5308ea0d89cf57d672c9d5950c323378e359677
9b43f030f28e51b74ee71b1eb4771760ffbe768f	b343b4ac4fef43e8b1b016620fc9b43068dd61fb
bbfa4d9c09686367b51862b23f998ad992c9057c	3af89243d5a74ffd84cf3934c7985f43cf6dc12f
76511d865a4035bbb6e911a0d9f9a549507a4e81	bc1e2dce8c40fe33face094a21f8cb4cfd2c403a
b5d6dd49a8daffdd6f2c632b2cd22dbb4a3f0abf	c3c5bcf73c6855f95277fc67a456a5762b08c1f8
c67249515c94081ed0ede2b1b88bd355d6acf795	a2f097b3b6b4a3b1a1d158eee7c913799e48ba0a
e1b3e853a42d04a183c60d80f0be842046474b6a	cfad0b4c8880e7397311797d1bb314149b8e056d
5c7f77600713bd8e8fac70ff8f355b6564612634	2cc7159557fec07cebe0861825919917bf62eb36
843bb7448da716fcb0a47512edac34e429ca29df	5ec3f42a45c0691565ff4459dc2cdea00e9d8cf4
981f73188a506cdf35b99926cd78d019d1c372d9	2ec7b9a7aa53132b2ba24c4e13e325ded5ac3c07
cc843eb9c6591077c503c1467a012e9817c90b17	784b22719fa17967d0bee432cabae8e6eea982ff

797e445ccc0552d3b1b08edc3e6e75dd3eb3b22f	a025e548b58fb62a6dafe60e0c08483ab2c8c565
ff885c2ac9845c52f6b41540762c11cf6240302d	d3d996e6e1d1c78bdd880fa278e5dcf095363cdd
d6ee63e837dc39c55963b5b16beffa3b9e51f55a	06da1adf20f269ccc6a67fab86707fd68b54a137
feec79b8373b3c7ccb8220ebf34fceae5c2dc053	24e919db5513f2149de652da04a442359cabdd63
d1ad26b5a4de174a8566b3f5afb00c8eaa282f62	9e0cdd05f4e37d0e0b834cf6518507427e097be4
ad815beff2fc2d90579494be8bb05714b845338f	38aff51bf65d2de77788684434fcea2d5ba86b64
74c44a2231b0bde9239a1297a66b92f407afaa09	32958744d3bd63a6c151d11a0d47fb953ea96036
b0fa6cd93d26ef747b74589377eb6b27f2c90d47	838bbe58c7e859ed588c19fd7fffc1b8566038c6
ce28eefb091b10452e6b601adc19ae8f82ce0078	d138d51cf5232d0d741bd81c9334d4e5b429d46f
bff922de90ce80b1cbf4902657d4dda6c8ab3dea	b91839445e9164d1dbf5bdc13e7f58ee0d5d02a5
981f7b1a4ad01bbd877dca1c25c5f3bd0df8089b	0c0a924378b077a053afc2b1d548f01a7791f833
7c0a690aeba573972073f8799b3d7e7b7406cddb	d2691aa3c2499004fa4e65911188f399e7eb5494
b4aee1b115577421ef2a16a72d7ca95cd178f817	64700bb0ce4ee281b6273c07b6d85fd6e77f85c4
630f1026a656203e51181603cc3e891f73452aee	ca325d0b2ded322038788c0d23348b1e312407c7
425e896245b12164235073c04472db5074062f29	cfc39658f81ad1a1509695b812e33f2e73114700
d29e9bdd613ebb7970e0621478fe8f365b7d1f86	db7a42c6701848c8c31228b664268b3066dc0779
84a8aa23cc86a2bf3e7e6560b285d027f4e36d83	1702d2e63ce0a4aecf9ec337cf130dd86c4bab6c
efc555aa2269c1f17443baaa3dd8cb20de7cadcb	1082f670ec7ff32f69715a0ee9d474a42c9d72b2
b550cd50401fdc6659748863e740e3ea263f8c8a	f2fd07b33365315ed930645ef98394e329742b7d
32185cda398ebd154753ee22d154b443d9780061	8131e9596cd97ca21a7d1da76ffa5f5964160e28
542b106d2d41e0c8068e095815a13b0b1070a084	e30401e341982cedea482e8f2bec4997597f441e
8992b57feee41f5469c9909656e43892760b4173	6381dafaaec79967d154f1a443dfdb09df2d65ee
34c150485a381c4a06e2995636e82b52097d5729	d7c842a8f78982506d1a52c0617080341b82b7d6
2e23f179e27b42a6e3c0b92107231fa5e96f5f0c	3c814bfd7a206016abed4f952bfff5c725b0941e
3507f7341709a8a53c63c513a9825867f5ef6218	b0a0985e8cd15dd99fa09f125839f62ab5866b4b
4a62a046fc5b4be8d51e40c5c1540f8aa0078d65	633380d5347decc10a15e62004b4f78cf61c4025
b15eece68f6a46a74977f44b37b202e82c841340	cc8a4728c6a796bab7ba48d153309d68ada981b8
ddf701726e076fb9285c8fa1cd64195d468faa9e	e5f427d1cedb05c4a2f6ab29e8a67e954c373ec6
728bf2cc514ed3fd4f4a76d47074c979b18811cd	a610766380d4ab8df074a0b956b0b2b1a7752a4a
495f7c399b2a92ce51a672978118d8e029a8674d	52db039a67e870caefc1c7ed314c368c4c95cfcd
674a00a4cada334decdfc692e835ae3a6a1d615d	c7963609c2d2fe89299de785654b004069936377
d1c8ffb5c605f5b48d9f6a4fe29952c618550ae8	e5b46e568fe0dd0c953962a15ee30a242e206cea
55e69739c65ca9e4aac1840e046ef6f9f1fc93d5	794a186039d6e1c38caf873b9743451dcd6b5633
50b8455340f12c4a8d15251cc153a1cfca558414	0957fee66c27ccde9d495155bc3b7029ecfe939a
24674c78202b51adf28b13ffe3c3845efb3956cc	37d2a4eba1056952df290e0bbee40f9ac15ccfa7
abc921a80344e33c60a630b7eadd1970127e8e0f	0323238fa311cacd343e5d2a4c5ffd85285a6ab7
281b820f96294280e6ffd47a53e709e0bf71b01e	00d891a35a6eb9ed86e268a9f0af7e0b4f2513e5
9de939acade304f3728f8756147e3b90e3c93ef3	6c1150e9caeee47555faf3359b1a8158c17fa2de
332dc11cad3a50207b9b2ef679e322ce4e70dcbd	555b2cc94c560152a0a6369434a84deafe447cff
a0eedb3ef04c61330fe4a7f8029b6d97f43ec369	e6319e1003ae382d2f80e43e41b4d11c7fd71c16
e8f0785c8206f9fd4e0595b8d71235e8329d63ee	a3389ec41eafbbe6540dbf4aa5ec9261ae4e3e4c
834e65ba633fa3d833448b0ed410e69f6d9c4e83	fe38d030f5ed60857582daf37ba7f2dbb3a2f639
c7b90dcdbce427b7e7b120aa59ca98a68a0b7477	eac8bab219e6c1075bf4e9c963e4f40458e61722
031b218556402716260f6c141f4652dbdced1849	ea7cbad9ecbb6806f2b50b293d523e1582358cd3

c231e4fae8afde3deaa16e0bb0c7105f8ca4bbd6	f4ace25a536ca5dc860d87bfcf71ec50f105c94f
c01abb304dd34df53354c740a13bb9d0cd37e54d	e4cf5f1a670feca669d738b2503d6daf6700c48f
8902e9879bfe83e622d0ea7394fa11cf48efda65	dac5403c1ff737498efae14fcea06292d3c2ba7a
1e0b906f6819eee9bd756b85899972f3ef4f8842	6ffbcdc353a3887bd60f05f5824646a4976e0dae
f7352d8b74c202b05257cfef2ef875f0fe28fa9c	adf457e076ec0f1c23a547512428c075d1bf18d7